IBM Systems - iSeries

Database
Distributed data management

Version 5 Release 4

IBM Systems - iSeries

Database
Distributed data management

Version 5 Release 4

Note

Before using this information and the product it supports, read the information in "Notices," on page 211.

Seventh Edition (February 2006)

This edition applies to version 5, release 4, modification 0 of IBM i5/OS (5722-SS1) and to all subsequent releases and modifications until otherwise indicated in new editions. This version does not run on all reduced instruction set computer (RISC) models nor does it run on CISC models.

Contents

Distributed data management

This topic contains i5/OS® distributed data management (DDM) concepts, information about preparing for DDM communications, and DDM-related programming information.

Although this topic collection does contain some information about systems other than iSeries™, it does not contain all the information that the other system types might need to communicate with the iSeries server using DDM. For complete information for a particular remote system type, refer to that system's documentation.

In this topic collection, the term DDM refers to the distributed data management architecture used by distributed data management (DDM) to define the protocols used for communicating between systems. DDM is also used to refer to the following items:

- Terms used to discuss DDM architecture (for example, DDM jobs, conversations, functions, requests, and commands)
- Source and target implementations of the DDM architecture
- DDM files used by DDM to access remote files
- Non-iSeries DDM products that support DDM (for example, System/36™, System/38™, and CICS/DDM)

This topic is intended for application programmers who are using i5/OS distributed data management (DDM) to prepare a system to access data in remote files and to control access to local files by remote systems.

Distributed Relational Database Architecture™ (DRDA®) also uses the DDM architecture.

Note: By using the code examples, you agree to the terms of the "Code license and disclaimer information" on page 209.

Related concepts

Distributed database programming

What's new for V5R4

This topic highlights the changes made to this topic collection for V5R4.

In V5R4, DDM support for the following functions:

- Distributed transaction processing (XA/JTA). (See JDBC distributed transactions for more information.)
- Server support for profile tokens. (See Security-related APIs for more information.)
- Protected conversations for RDB DDM files over TCP/IP.

How to see what's new or changed

To help you see where technical changes have been made, this information uses:

- The » image to mark where new or changed information begins.
- The « image to mark where new or changed information ends.

To find other information about what's new or changed this release, see the Memo to users.

Printable PDF

Use this to view and print a PDF of this information.

To view or download the PDF version of this document, select Distributed data management (about 2903 KB).

Saving PDF files

To save a PDF on your workstation for viewing or printing:

1. Right-click the PDF in your browser (right-click the link above).
2. Click the option that saves the PDF locally.
3. Navigate to the directory in which you want to save the PDF.
4. Click **Save**.

Downloading Adobe Reader

You need Adobe Reader installed on your system to view or print these PDFs. You can download a free copy from the Adobe Web site (www.adobe.com/products/acrobat/readstep.html) .

Introduction to i5/OS DDM

This topic describes the purpose of distributed data management (DDM), the functions that DDM supplies on the iSeries server, and the concepts of i5/OS DDM.

DDM is part of the i5/OS licensed program. i5/OS DDM as a source supports Level 2.0 and below of the DDM architecture. i5/OS DDM as a target supports Level 2.0 and below for *record file* (a file on disk in which the data is read and written in records) types and Level 3.0 and below of the DDM architecture for stream files (documents) and directories (folders).

The DDM support on the iSeries server allows application programs or users to access data files that reside on remote systems, and also allows remote systems to access data files on the local iSeries server, as shown in Figure 1 on page 3. Any system that supports the DDM architecture as a source system can access data (if authorized to do so) on any other system to which it is attached. The attached system must support DDM as a *target system* (the system that receives a request from another system to use one or more files located on the system). However, the source and target systems must support compatible subsets and levels of the DDM architecture.

The folder management services (FMS) support allows personal computer users to access folders and documents that reside on an iSeries target server. Remote systems that support Level 3.0 or Level 2.0 of the DDM architecture for the stream access method can access folders and documents on the local iSeries server.

DDM extends the file accessing capabilities of the iSeries server database management support. In this topic collection, *database management* refers to the system function that controls **local** file processing; that is, it controls access to data in files stored on the local iSeries server, and it controls the transfer of that data to requesting programs on the same server.

Distributed data management controls **remote** file processing. DDM enables application programs running on one iSeries server to access data files stored on another server supporting DDM. Similarly, other systems that have DDM can access files in the database of the local iSeries server. DDM makes it easier to distribute file processing between two or more servers.

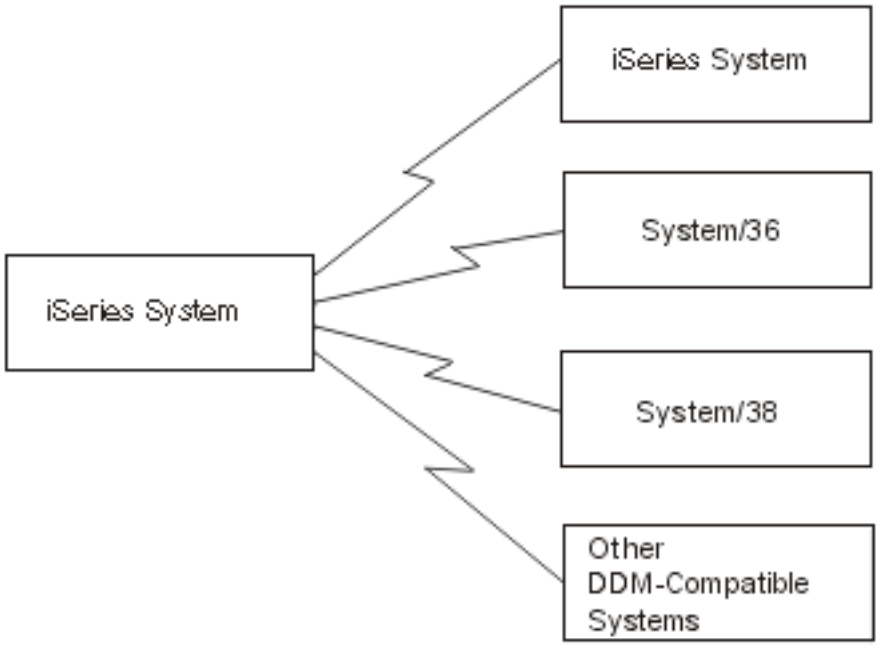


Figure 1. Source and target systems

Systems that use DDM communicate with each other using the Advanced Program-to-Program Communication (APPC) support, Advanced Peer-to-Peer Networking® (APPN) support, or TCP/IP. See the *Communications Management* manual on the V5R1 Supplemental Manuals Web site and the APPC, APPN, and HPR topic for information needed to use APPC and APPN.

Folder management services (FMS) allows local access to documents or folders that are on the iSeries server. Personal computers might access folder management functions on the server by using DDM.

Note: Distributed data management for the IBM® Personal Computer uses the iSeries portion of the iSeries Access Family licensed program.

As shown in Figure 2 on page 4, the server on which a user application issues a request involving a remote file is called a *source system*. The server that receives the request for one of its files is called the *target system*. A system can be both a source and target system for separate requests received at the same time.

Using DDM, an application program can get, add, change, and delete data records in a file that exists on a target system. It can also perform file-related operations, such as creating, deleting, renaming, or copying a file from the target system to the source system.

When DDM is in use, neither the application program nor the program user needs to know if the file that is needed exists locally or on a remote system. DDM handles remote file processing in essentially the same way as local file processing is handled on the local system, and the application program normally does not receive any indication of where the requested file is located. (However, in error conditions, messages are returned to the user that indicate, when necessary, that a remote system was accessed.) Informational messages about the use of target system files are included in the source system's job log.

When DDM is to be used, only application programmers need to know where the file is located and, using control language (CL) commands outside of the high-level language (HLL) programs, they can

control which file is used. However, the programmers can also choose to use specific recovery functions to handle certain communications failures; the HLL programs might need to be changed to include handling any such failure.

Therefore, iSeries BASIC, ILE COBOL, ILE RPG, ILE C, and iSeries programs that are compiled to process database files on the local server might not need to be changed or recompiled for DDM to process those same files when they are moved to or exist on a remote server.

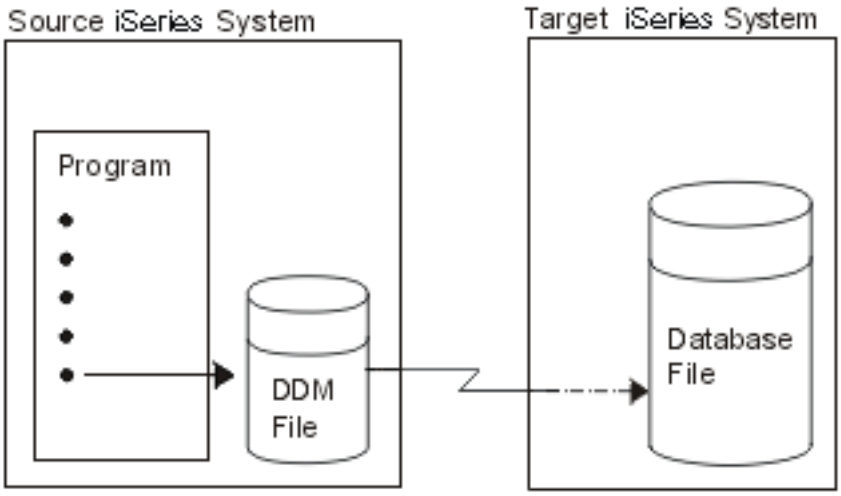


Figure 2. Move a program from a source to a target system

Related concepts

"Prepare to use DDM" on page 43
There are several requirements that must be met for DDM to be used properly.

"Use language, utility, and application support for DDM" on page 24
This topic describes the language, utility, and application program support that is provided on the iSeries server for DDM.

System compatibility

DDM can be used to communicate between systems that are architecturally different.

For example, although the architectures of the iSeries server and System/36 are different, these systems can use DDM to access files in each other's database. To successfully communicate with each other, each system must have an implementation of DDM that is compatible with Level 2.0 or below of the IBM DDM architecture. Also, each type of system might use all or only part of the IBM DDM architecture or might have extensions to the architecture.

If you are communicating with any non-iSeries servers, you must consider the level of DDM support provided by those servers for such things as unique security considerations.

For a list of the DDM architecture manuals that supply the details about Level 3.0 or below of the IBM DDM architecture, see Related information for distributed data management.

Related concepts

"Security" on page 48
This topic describes how iSeries security relates to DDM, and how it can limit access to the data resources of a target server by source server programs and users.

Related reference

"Related information for distributed data management" on page 208
Listed here are the product manuals, Web sites, and information center topics that relate to the distributed data management topic. You can view or print any of the PDFs.

Overview of DDM functions

This topic gives an overview of the types of DDM functions that can be done on a target server.

The following *file* operations, normally specified in **HLL programs**, can be done on files at target servers:

- Allocating, opening, or closing one or more files.
- Reading, writing, changing, or deleting records in a file.

The following *file* and *nonfile* operations, normally specified in **CL programs** or by CL commands, can be done on files at the target servers:

- Copying the contents of a file.
- Performing operations on physical or logical file members (such as adding, clearing, or removing members), but only if the target is an iSeries server or System/38.
- Accessing remote *files* for nondata purposes, such as:
  - Displaying information about one or more files, using commands such as Display File Description (DSPFD) and Display File Field Description (DSPFFD). These commands can display the file attributes of the DDM file on the source system or the file or field attributes of the remote file on the target system.
  - Controlling the locking of files on the target system, using the Allocate Object (ALCOBJ) and Deallocate Object (DLCOBJ) commands.
  - Deleting, renaming, creating, and changing files using the Delete File (DLTF), Rename Object (RNMOBJ), Create Physical File (CRTPF), Create Source Physical File (CRTSRCPF), Create Logical File (CRTLF), Change Physical File (CHGPF), Change Logical File (CHGLF), and Change Source Physical File (CHGSRCPF) commands.
- Accessing remote *systems* for non-data purposes:
  - Send a CL command to the target system (an iSeries server and a System/38 only) so it can be run there, instead of on the source system (where it might not be useful to run it), using the Submit Remote Command (SBMRMTCMD) command. The SBMRMTCMD command is the method you use to move, save, or restore files on a target server. For example, a Move Object (MOVOBJ) command might be sent to move a database file on the target server. (For typical uses of the SBMRMTCMD command, refer to its description in Use CL and DDS with DDM or refer to the CL topic for a more complete description.)

Various other nonfile-related operations can also be done on the target server.

Related concepts

Control language

"Use CL and DDS with DDM" on page 66
This topic contains DDM-related information about specific iSeries control language (CL) commands, data description specifications (DDS) considerations, DDS keywords, and DDM user profile authority.

Basic DDM concepts

This topic collection gives the basic concepts of DDM.

Because remote file processing is much like local file processing, these topics should provide sufficient conceptual information for most users of DDM. Another topic provides additional, more detailed concepts, and the topic Additional DDM concepts is intended primarily for the experienced programmer who wants or needs to know more about DDM.

From an user's viewpoint, accessing data on a remote system is much the same as accessing data on the local system. The main difference is the additional time needed for the data link to pass the data between the systems whenever the remote file is accessed. Otherwise, the user or application program does not need to know whether the data being accessed came from a local or remote file. Refer to Performance considerations for DDM for additional considerations.

For DDM iSeries-to-iSeries file processing, remote file processing is done much the same as local file processing. The purpose of this topic collection is to describe the things that are different for DDM. Also, because other systems can use DDM, those considerations and concepts are covered as needed to enable the iSeries programmer to successfully prepare the server for using DDM.

The DDM concepts in these topics describe mainly iSeries-to-iSeries remote file processing. For purposes of illustration, concepts that relate to System/36 and System/38 are shown in some examples. If you are using DDM on both System/36s and iSeries servers, you should be aware that the concepts for both types are similar, except in the way they point to the remote file: An iSeries server and a System/38 use a separate *DDM file* to refer to each remote file to be accessed; System/36 uses a network resource directory that contains one *network resource directory entry* for each remote file to be accessed.

Note: Although DDM supports other functions besides opening and accessing remote files, the concepts described in this topic collection deal primarily with remote file accessing.

Related concepts

"Additional DDM concepts" on page 13
Most users of DDM will not need the information in the remainder of these topics; it is intended primarily for experienced programmers who need to know more about DDM.

"Performance considerations for DDM" on page 134
These topics provide information to help you improve performance when using DDM and also provide some information about when to use something other than DDM to accomplish some functions.

Parts of DDM

DDM consists of several parts to handle remote file processing among the systems using DDM.

- Source DDM (SDDM)
- Target DDM (TDDM)
- DDM file

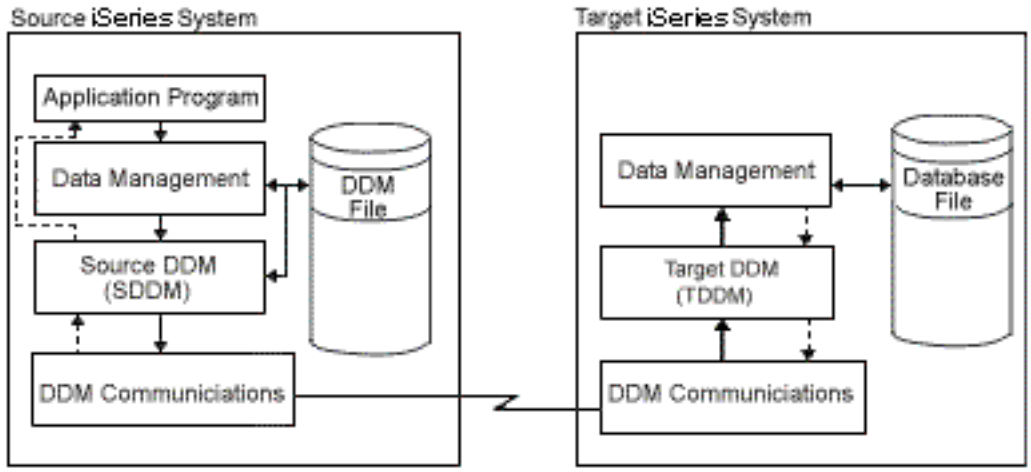


Figure 3. Communicate with DDM

The preceding figure shows how the basic parts involved in DDM communications on both systems relate to each other.

When a DDM file is accessed by a source system user or program, a DDM conversation is started between SDDM and TDDM for the job in which the program or user is operating.

Parts of DDM: Source DDM

The support on the source (or local) iSeries server is started, as needed, within a source job to do DDM functions.

The source DDM (SDDM) translates requests for remote file access from source server application programs into DDM requests that are routed to the target server for processing. The SDDM support establishes and manages a DDM conversation with the target server that has the desired remote file.

When an application program first attempts to access a remote file, a search for the requested DDM file is done on the source server. As with local file processing, if the file name is not qualified with a library name, the current library list for the job in which the program is running is searched for the specified file. When the file is found, the server accesses the file, determines that it is a DDM file and starts the SDDM.

When the SDDM is started, it checks to see if a DDM conversation is already active between the source job starting the SDDM and the target server identified by the remote location and mode values in the DDM file. If a conversation that can be used exists, it is used. If not, a program start request is issued to the appropriate target server to start a TDDM (a target job) on the target server to establish a DDM conversation between the SDDM and TDDM. Parameters that are automatically created from information in the DDM file about the remote file are passed when the remote server sends a program start request.

After the TDDM is started, the SDDM can forward each program request to the target job for processing. If, for example, input/output (I/O) operations are to be done on a remote file, the program opens the file and then issues the desired operation requests. The SDDM forwards the open request and the TDDM opens the remote file. Then the SDDM forwards each file operation request to the TDDM, and both of them handle the interchange of data between the application program and the remote file. When a DDM function is being processed, the requesting program waits for the function to be completed and the results to be received, just as it does for local file operations.

Related concepts

"iSeries server as the source server for DDM" on page 13
When an application program or user in a source server job first refers to a DDM file, several actions occur as part of processing the request on the source server.

Parts of DDM: Target DDM

A target server job is started on the target (or remote) server as a result of an incoming DDM request and ends when the associated DDM conversation ends.

The target DDM (TDDM) translates DDM requests for remote file access into data management requests on the target server and then handles the return of the information that is to be sent to the source server.

The TDDM is started when the remote server sends a program start request. The TDDM is started as a batch job on the target server. After the TDDM is started and a DDM conversation is established, the TDDM waits for a request (such as a file open or read operation, or a nonfile-related operation) to be sent by the SDDM.

When the TDDM receives a request to access an object on the target server, it searches for the requested object. If the object was not qualified with a library or path name, the current library list or current directory for the target job is searched.

When the requested object is found, the TDDM passes the first operation requested to database or folder management on the target server, which performs the operation on the object. When the operation is completed, database or folder management services return the results of the operation to the TDDM, which passes it to the SDDM. The SDDM passes the results and any accompanying data (such as records requested on a read operation) to the application program. These actions are repeated for each subsequent I/O operation request received, until the object is closed. If an operation does not complete successfully, the SDDM returns an error message to the program, providing information about the error.

The TDDM and the target job remain active until the DDM conversation is ended by the source server job that started it.

Related concepts

"iSeries server as the target server for DDM" on page 18
The iSeries target DDM (or TDDM) is actually a job that runs a DDM-related target server program. It is started when the source server sends a program start request (an SDDM).

Parts of DDM: DDM file

A system object with type *FILE exists on the source server to identify a remote file. It combines the characteristics of a device file and a database file. As a device file, the DDM file refers to a remote location name, local location name, device name, mode, and a remote network ID to identify a remote server as the target server. The DDM file appears to the application program as a database file and serves as the access device between a source server program and a remote file.

A DDM file is a file on the source server that contains the information needed to access a data file on a target server. It is *not* a data file that can be accessed by a program for database operations. Instead, when a source server program specifies the name of a DDM file, the file information is used by DDM to locate the remote file whose data *is* to be accessed.

DDM file information is based on *locations*. The remote location where the remote file is located is specified using the remote location name (RMTLOCNAME) parameter on the Create DDM File (CRTDDMF) or Change DDM File (CHGDDMF) commands.

The remote file name specified on the CRTDDMF or CHGDDMF commands must be in the format used by the remote system.

Another use of the DDM file is to submit control language (CL) commands to the target system to run on that system. In this case, the remote file normally associated with the DDM file is ignored.

Related reference

"SBMRMTCMD (Submit Remote Command) command" on page 69
The Submit Remote Command (SBMRMTCMD) command submits a command using DDM to run on the target server.

Create a DDM file using SNA:

You can create a DDM file that uses SNA as the communication protocol for connecting with the remote system.

Each DDM file that uses SNA contains the following information.

DDM file value and description of values

DDM file name
The name of the DDM file on the source system that is used to identify a specific remote file.

Remote file name
The actual file name of the remote file; that is, the name by which it is known on the target server. (For a target System/36, this is the file *label* of the remote file.)

Remote location name
The name of the remote location where the remote file exists. This remote location name provides the data link to the target server (remote location) by using APPN/APPC, over which a DDM conversation is established when this DDM file is accessed.

Device
The name of the device on the source server used to communicate with the remote location.

Local location name
The name of the local location. This is the name by which the target server knows your server. Your server can consist of more than one local location.

Mode The name of the mode to be used to communicate between the local location and remote location.

Remote network ID
The remote network ID to be used with the remote location. This value further qualifies the remote location name. Two locations with the same remote location name but different remote network IDs are viewed as two distinctly separate locations.

Type The type of connection to be used to communicate with the remote location when the DDM conversation is established with the remote server. To create a DDM file that uses an SNA connection, specify *SNA. This is the default type.

Create a DDM file using TCP/IP:

You can create a DDM file that uses TCP/IP as the communication protocol for connecting with the remote server.

Each DDM file that uses TCP/IP contains the following information.

DDM file value and description of values

DDM file name
The name of the DDM file on the source server that is used to identify a specific remote file.

Remote file name
The actual file name of the remote file; that is, the name by which it is known on the target server.

Remote location name
The name of the remote location where the remote file exists. This remote location name provides the data link to the target server (remote location) by using TCP/IP, over which a DDM conversation is established when this DDM file is accessed.

Type The type of connection to be used to communicate with the remote location when the DDM conversation is established with the remote server. To create a DDM file that uses TCP/IP, specify *IP.

Related concepts

"Manage the TCP/IP server" on page 124
These topics describe how to manage the DRDA and DDM server jobs that communicate using sockets over TCP. It describes the subsystem in which the server runs, the objects that affect the server, and how to manage those resources.

Create a DDM file using RDB directory entry information:

You can create a DDM file that uses the remote location information from a relational database (RDB) directory entry.

Each DDM file that uses an RDB directory entry contains the following information.

DDM file value and description of values

DDM file name
The name of the DDM file on the source server that is used to identify a specific remote file.

Remote file name
The actual file name of the remote file; that is, the name by which it is known on the target server.

Remote location name
Specify *RDB to indicate that the remote location information is taken from an RDB directory entry.

Relational database
The name of the relational database entry used for the remote location information. The remote location information in the RDB directory entry is used to establish the data link to the target server (remote location), over which a DDM conversation is established when the DDM file is accessed.

You need to specify an RDB directory entry associated with an auxiliary storage pool (ASP) group for the DDM file's remote location information to access that ASP group.

Related concepts

Disk management

Distributed database programming

Effect of job description on ASP group selection:

When the target DDM server is configured to use ASP groups, and the DDM file specifies a relational database name, the relational database entry specified in the DDM file on the client is used to establish the ASP group for the target job.

When using a DDM file that does **not** specify a relational database name, the target job's ASP group is established using the initial ASP group attribute in the job description for the user profile that the target job is running under.

Example: Use the basic concepts of DDM in an APPC network:

This topic presents a sample application that uses DDM to access a remote file.

The application can be run by a company that has warehouses located in several cities. The following figure illustrates the relationships among the primary items included in a DDM file.

On an iSeries server in Chicago, an Open Database File (OPNDBF) command requests that file CUST021 be opened for input. Because the file name was not qualified on the command, the library list for the source job is used to find the file, which is stored in the NYCLIB library.

Because CUST021 is a DDM file, the SDDM on the CHICAGO server is started in the source job when the file is opened. The SDDM uses the remote location and mode names (NEWYORK and MODENYC) from the DDM file to establish a DDM conversation with and start a target job (TDDM) on the appropriate target server (NEWYORK). The remote file to be accessed by the source server program is CUSTMAST in library XYZ.

The TDDM receives the remote file name from the SDDM and then allocates and opens the file named CUSTMAST, which corresponds to the DDM file named CUST021 on the source server.

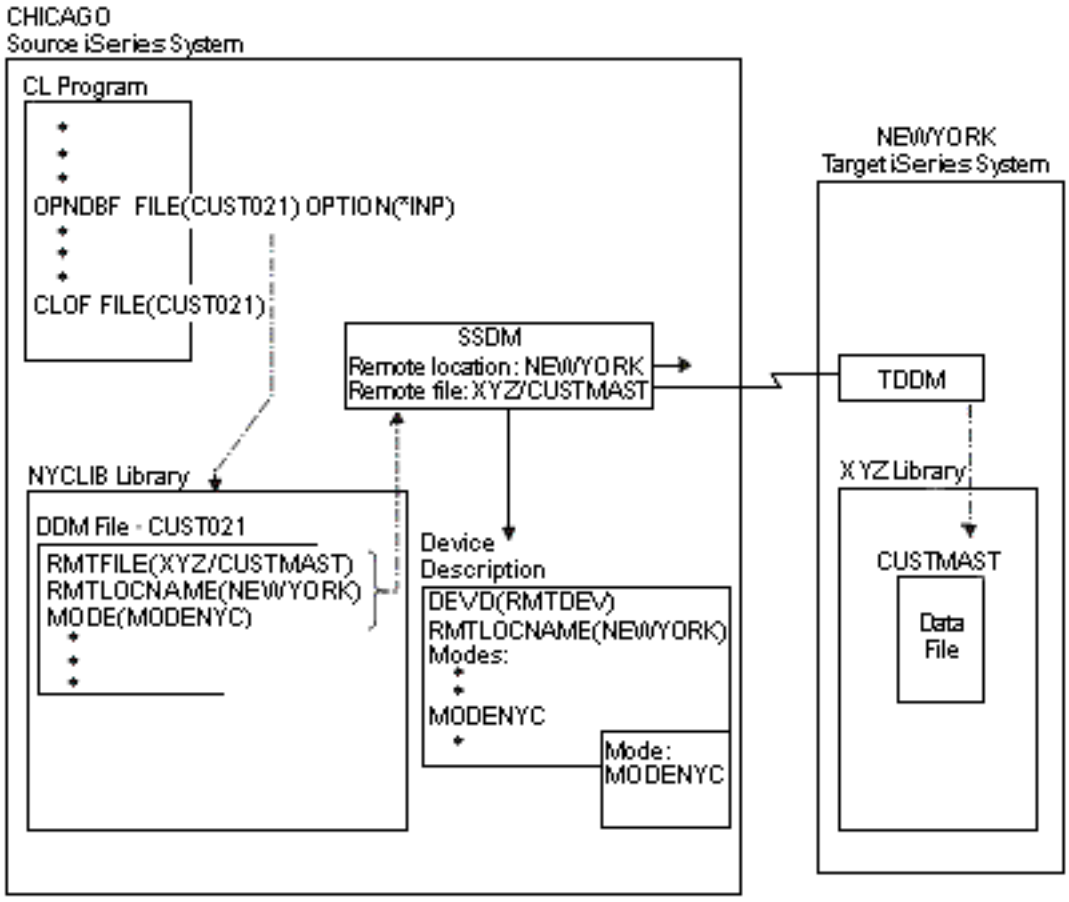


Figure 4. Relationships among DDM file parameters and the systems

The remote location name in the DDM file identifies the remote server where the file exists. The local server uses the remote location name as well as other values specified in the DDM file to select a device description. The device description can be either manually created or, if APPN is being used, automatically created and activated by the server. The SDDM establishes a DDM conversation with the target server using the values NEWYORK and MODENYC in the APPC remote location name. The

APPC-related support must have been started on the target server before the request is issued by the SDDM. (No special support is required on the source server.)

Note: The APPN parameter on the Create Controller Description (APPC) (CRTCTLAPPC) and Create Controller Description (SNA Host) (CRTCTLHOST) commands determines whether the APPN support is used.

Related concepts

APPC, APPN, and HPR

Example: Use the basic concepts of DDM in an APPN network:

The Advanced Peer-to-Peer Networking (APPN) support of an iSeries server can be used to allow DDM access to systems not directly connected to the local server.

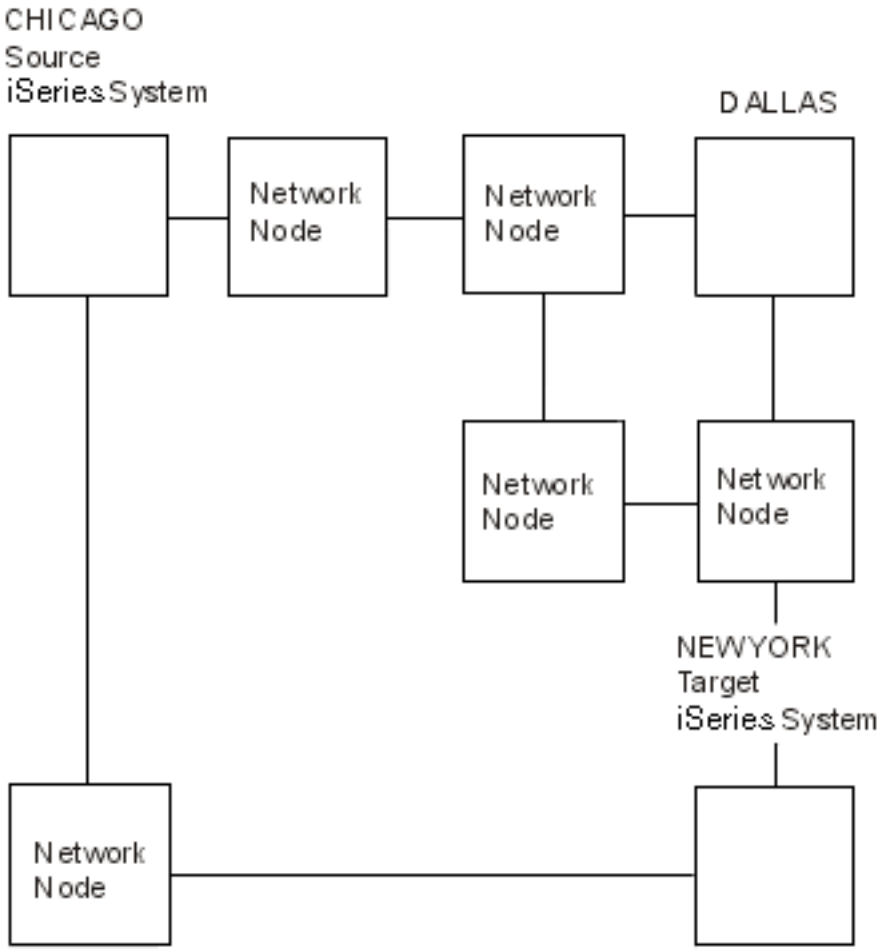


Figure 5. Use DDM in an APPN network

Figure 1 in "Example: Use the basic concepts of DDM in an APPC network" on page 10 shows a program on the Chicago server accessing a file on the New York server. Although the servers are shown as directly connected, the same DDM concepts apply if the network is configured as shown in the preceding figure. When the DDM file CUST021 in the figure is opened on the Chicago server, the APPN support finds the remote location named NEWYORK, determines the optimal path through the network, and establishes a DDM conversation with that location. Although there might be several other servers (network nodes)

forwarding the data between CHICAGO and NEWYORK, the source DDM and target DDM function as if there were a direct connection between these two servers.

If the file CUSTMAST were moved from NEWYORK to some other server in the network (for example, DALLAS), then in this example, the DDM file at CHICAGO needs to be changed. The remote location name would be changed from NEWYORK to DALLAS. If a large number of servers in the network refer to the file CUSTMAST, then movement of the file results in a change to the DDM file at each of these servers. By using the iSeries capability to have multiple local location names, maintenance of these files is reduced.

In the preceding figure, the server NEWYORK can be given two local location names, NEWYORK and FILELOC. The DDM file at CHICAGO uses FILELOC as the remote location name. When access to file CUSTMAST is required, APPN finds the location FILELOC in the system named NEWYORK, and the DDM conversation is established as before.

If the file CUSTMAST is now moved from NEWYORK to DALLAS, the user at NEWYORK deletes the local location FILELOC from his server, and it is added to the server at DALLAS. This is done by using the APPN local location list. When the program in CHICAGO now attempts to access the file CUSTMAST, the APPN support finds the remote location FILELOC at the server in Dallas, and the DDM conversation is established to that server. The movement of CUSTMAST did not result in a change to the DDM file at CHICAGO.

This example shows the concept of multiple local locations and how reduced maintenance results when files are moved from one server to another. The example is not intended to suggest that a unique location name should be used for every file accessed through DDM. The decision of which files should be associated with separate local locations should be based on such factors as the movement of these files and the number of remote servers accessing these files.

Additional DDM concepts

Most users of DDM will not need the information in the remainder of these topics; it is intended primarily for experienced programmers who need to know more about DDM.

Described are conceptual details and examples about:

- Program start requests, which start the TDDMs (target jobs)
- Open data paths (ODPs), used to access the files
- Remote location information
- DDM conversations, established for source and target communications
- Source and target jobs
- I/O operations within a job

Related concepts

"Operating considerations for DDM" on page 111
This topic provides task-oriented information and examples that describe various aspects of DDM operation considerations.

iSeries server as the source server for DDM

When an application program or user in a source server job first refers to a DDM file, several actions occur as part of processing the request on the source server.

All of these actions, as well as those required on the target server, must complete successfully before any operations (file or nonfile) requested by the source program can be done. When the DDM file is referred to:

- If the request is to open a file, its information is used simultaneously to create an open data path (ODP) on the source server and to start the SDDM support, which runs within the same job as the

source program. The SDDM also uses the information: to convert the source server request into a DDM request, to communicate with the appropriate target server, and to establish a DDM conversation to be used for the source job. (The ODP is partially created with the DDM file information; it is not usable until the SDDM processes the remaining information after the DDM conversation is established.)

- The communications portion of DDM establishes a communications path with the target server. The target *server* is identified by using the remote location information specified in the DDM file, and the target *file* is identified by the remote file name. Other information about e remote location, not kept in the DDM file, is stored by the SDDM. This includes the transaction program name, user ID, activation group number, and scope of the conversation. Using the remote location information, the TDDM is started on the target server and a DDM conversation is established when the remote server receives the program start request. The conversation is established the first time the remote file is accessed, but only if a conversation using the same remote location values for that target server does not already exist for the source job.
- After the DDM conversation is established, the SDDM (which can be used by multiple programs and multiple DDM files in the same source job) sends the DDM architecture command to the TDDM, for file-related requests. This command describes the file operation to be done and contains the name of the remote file (specified in the DDM file) to be accessed. For nonfile-related requests, such as when the Submit Remote Command (SBMRMTCMD) command is used, the remote file name is not sent to the TDDM; the remote file name is ignored.

The SDDM converts each program request for a file open or input/output operation (received by using the DDM file and ODP) into an equivalent DDM command request and then sends it to the target server.

The following figure shows the basic parts on the source iSeries server that are involved in accessing remote files.

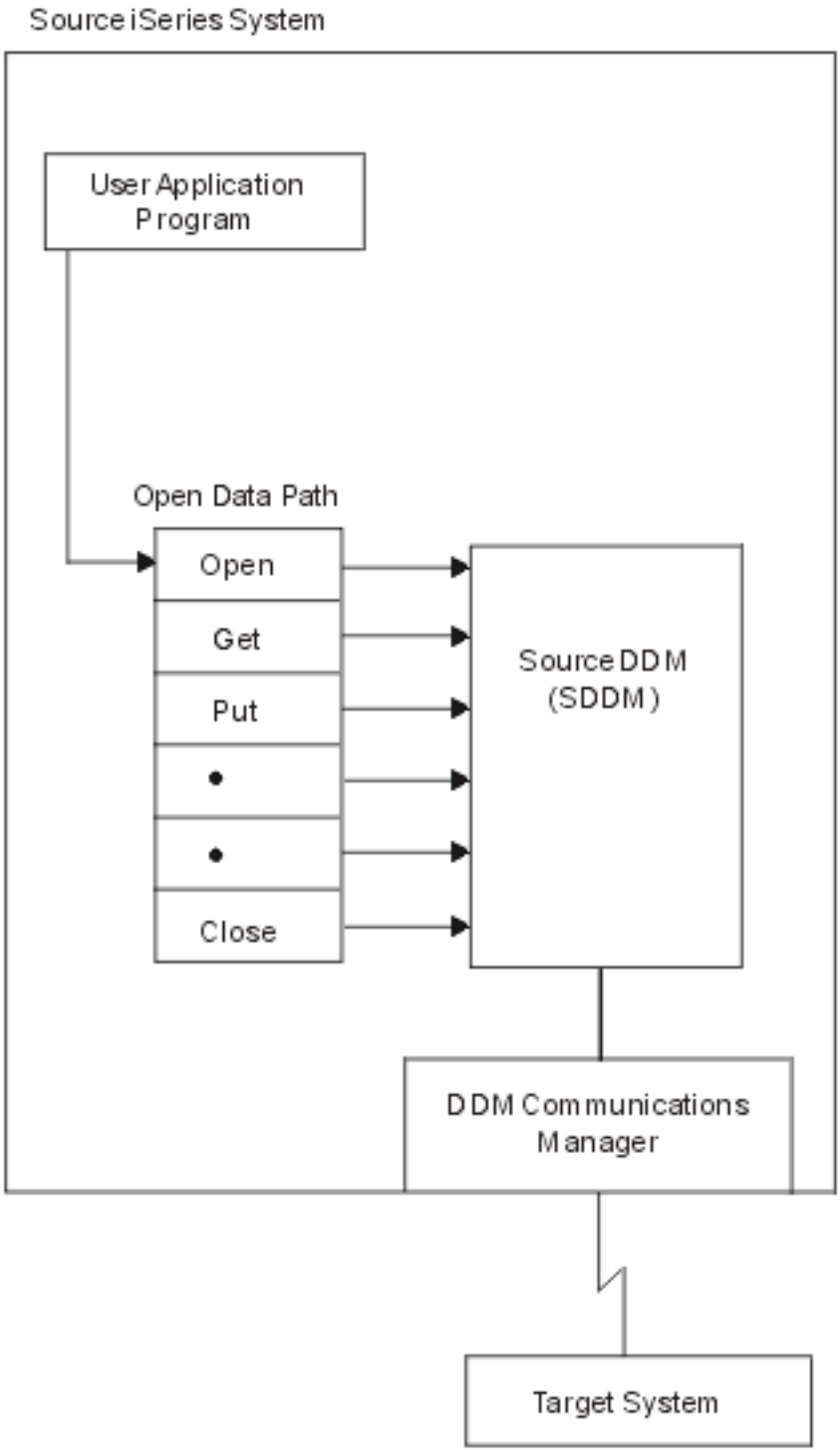


Figure 6. iSeries server as the DDM source server

After each request is handled by the target job, the DDM response from the target server is returned, converted by the SDDM into the appropriate form, and passed back to the user. The response might include data (if data was requested) or an indication of status (for other types of file access). The source program waits until the function completes and the results are received.

The following figure shows a simplified example of the interchange of data between the source and target servers for a typical request to access a remote file.

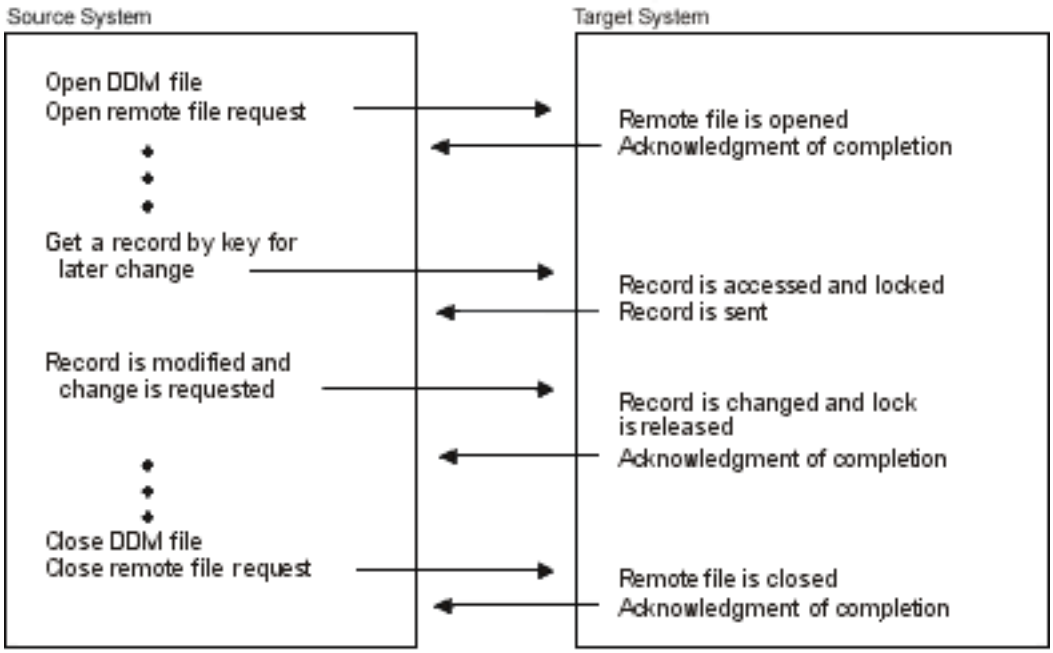


Figure 7. Typical handling of an I/O operation request

After the first DDM file that was opened in the job is closed, the DDM conversation that it used is normally kept active. This allows the same program or another program in the job to use the same conversation when opening another DDM file, or doing other DDM-related operations. (For example, in Figure 9 on page 21, source job 3A has two DDM files using the same conversation.) This saves the time and resources required to establish a new conversation every time a new DDM file that uses the same remote location information is used in that job.

When a DDM file is closed, the DDM conversation remains active, but nothing happens in the conversation until the SDDM processes the next DDM-related request from a program. While it is not being used, however, the conversation can be dropped. This can occur if the DDMCNV job attribute's default value of *KEEP is changed to *DROP using the Change Job (CHGJOB) command, or if the Reclaim DDM Conversations (RCLDDMCNV) command or Reclaim Resources (RCLRSC) command is used while the job is active.

Related concepts

"Parts of DDM: Source DDM" on page 7
The support on the source (or local) iSeries server is started, as needed, within a source job to do DDM functions.

"Use CL and DDS with DDM" on page 66
This topic contains DDM-related information about specific iSeries control language (CL) commands, data description specifications (DDS) considerations, DDS keywords, and DDM user profile authority.

Related reference

"DDMCNV parameter considerations" on page 97
The DDMCNV parameter is a job-related parameter that controls whether Advanced Program-to-Program Communication (APPC) or iSeries conversations in the job that is allocated for DDM use (that is, DDM conversations) are to be automatically dropped or kept active for the source job.

"Control DDM conversations" on page 120
Normally, the DDM conversations associated with a source server job are kept active until one of the conditions described in this topic is met.

Integrated Language Environment and DDM:

Integrated Language Environment® (ILE) introduces the concept of activation groups that run within jobs on the iSeries server.

An *activation group* is a substructure of a runtime job. It consists of server resources (storage for program or procedure variables, commitment definitions, and open files) allocated to one or more programs. An activation group is like a miniature job within a job. By default, all DDM conversations are scoped to the activation group level. To *scope* is to specify the boundary within which server resources can be used. Programs that run in different activation groups start separate DDM conversations when they use the same DDM file or the same remote location information. Sharing of existing DDM conversations takes place within the confines of the activation group. A DDM conversation can be scoped to the job level by specifying OPNSCOPE(*JOB) on the OPNDBF command.

Related information

 ILE Concepts PDF

Source server actions dependent on type of target server:

If the target server is not another iSeries server or System/38, only the DDM architecture commands defined in Level 2.0 and earlier of the DDM architecture are used.

If the target is an iSeries server or a System/38, then iSeries server and System/38 extensions to the architecture are used to support some operations not defined by the Level 2.0 DDM architecture. Examples of System/38 and iSeries extensions to the architecture are the Submit Remote Command (SBMRMTCMD) and processing file members of remote files. For creating a file when the source is an iSeries server and the target is also an iSeries server, an iSeries extension is used.

Target servers that are not iSeries servers or System/38s might not be capable of handling all of the functions that an iSeries server or a System/38 can handle. For example, a System/36 does not support relative record processing and keyed record processing with one open operation; therefore, programs that mix accessing records in a file by key or relative record do not work if the file is on a System/36. In addition, target servers that do not support Level 2.0 of the DDM architecture can only handle functions defined in the level they support.

Neither the System/36 nor the System/38 support access to folder management objects.

Note: An iSeries server only allows access to folder management services (FMS) objects when the source supports Level 2.0 of the DDM architecture for *stream files* (files on disks in which data is read and written in consecutive fields without record boundaries) and directories, for example, the IBM Personal Computer using DDM.

An iSeries server as a source server does not support access to stream files and directories.

Related concepts

"Use language, utility, and application support for DDM" on page 24
This topic describes the language, utility, and application program support that is provided on the iSeries server for DDM.

Related reference

"SBMRMTCMD (Submit Remote Command) command" on page 69
The Submit Remote Command (SBMRMTCMD) command submits a command using DDM to run on the target server.

iSeries server as the target server for DDM

The iSeries target DDM (or TDDM) is actually a job that runs a DDM-related target server program. It is started when the source server sends a program start request (an SDDM).

For source iSeries servers, the program start request is started on the source server using information contained in the IBM-supplied intersystem communications function (ICF) file for DDM. The remote location information in the DDM file being accessed is used to send the program start request to the appropriate target server.

The attributes of the target job are determined by the values specified on the Add Communications Entry (ADDCMNE) command, which is used on the target server to add a communications entry to the subsystem description used for the job. This command identifies the device description, the job description (including the library list for the target job), and the default user profile to be used by the subsystem.

For an iSeries Access Family connection, the routing entry in the QIWS subsystem for DDM (CMPVAL ('DDM')), along with the device description the personal computer is connected to, is used to obtain the attributes of the target job.

After it is started, the TDDM does the following things:

- For database files:
  - Handles communications with the source system by using a DDM conversation established over an APPC, over TCP/IP, or over an iSeries Access Family data link.
  - Converts the access requests from the source server into the equivalent iSeries functions and runs them on the target server. After the target object is located, the target server-created ODP and target database management services are used to access the object for whatever operation is requested. The TDDM can, for example, pass requests that open the object and then do requested I/O operations to the objects.
  - Includes iSeries or System/38 extensions to the DDM Level 2.0 architecture for requests received from the source server (if the source is an iSeries server or a System/38), which allow most iSeries functions that operate on local servers to also work on remote iSeries servers. For example, it might receive a SBMRMTCMD command from the source server (an iSeries server or a System/38) to do a nonfile-related operation, such as using the CL command Replace Library List (RPLLIBL) to replace the library list within the current target job.
  - Converts target iSeries responses to the equivalent DDM responses and sends them back to the source server. When the source server is an iSeries server or System/38, the actual iSeries or System/38 messages are sent back to the source server.
- For folder management services objects:

  Converts the DDM stream and directory access requests into the equivalent iSeries folder management services functions and then runs them on the target server. The following commands are supported:
  - Change Current Directory (CHGCD)
  - Change File Attributes (CHGFAT)
  - Close Directory (CLSDRC)
  - Close Document (CLOSE)
  - Copy File (CPYFIL)
  - Create Directory (CRTDRC)
  - Create Stream File (CRTSTRF)
  - Delete Directory (DELDRC)
  - Delete File (DELFIL)
  - Force Buffer (FRCBFF)
  - Get Data Stream (GETSTR)

- Get Directory Entry (GETDRCEN)
- List File Attributes (LSTFAT)
- Load Stream File (LODSTRF)
- Lock Data Stream (LCKSTR)
- Open Directory (OPNDRC)
- Open Document (OPEN)
- Put Data Stream (PUTSTR)
- Query Current Directory (QRYCD)
- Query Space Available (QRYSPC)
- Rename Directory (RNMDRC)
- Rename File (RNMFIL)
- Unload Stream File (ULDSTRF)
- Unlock Data Stream (UNLSTR)

The following figure shows the basic parts on the target iSeries server that are involved in processing the requested destination file.

The TDDM runs as a separate batch job, just as any other user APPC, TCP/IP, or iSeries Access Family target application. A new TDDM, using additional target server resources, is started for each distinct source server program start request received by the target server. There is one target job for each DDM conversation. Each TDDM can handle access requests for multiple files in the DDM conversation.

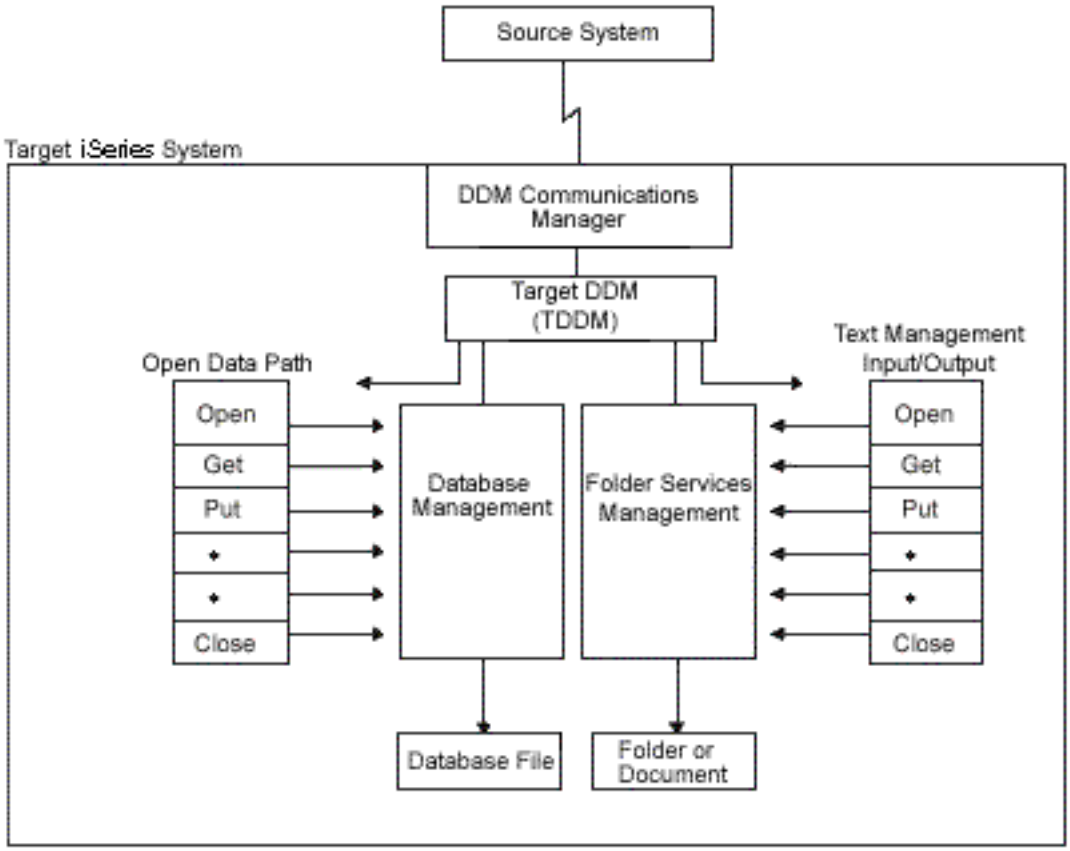


Figure 8. iSeries server as the DDM target system

The subsystem, user profiles, and server resources to be used by the TDDM are defined the same as they are for other types of jobs.

Related concepts

"Parts of DDM: Target DDM" on page 8
A target server job is started on the target (or remote) server as a result of an incoming DDM request and ends when the associated DDM conversation ends.

DDM-related jobs and DDM conversations

This topic provides additional information about activation groups, source server jobs, target server jobs, and the DDM conversations used by those jobs.

For remote file processing, at least two separate jobs are used, one running on each server: a source job and a target job. (The source server job is the one in which the user application is running.) Multiple application programs can be running in different activation groups within a single source job. Each activation group within a source job has a separate DDM conversation and target job for the remote location information specified in the DDM files. Multiple DDM files share a conversation when the following items are true:

- The files are accessed in the same activation group within a source job.
- The files specify the same remote location combination.

For each DDM conversation, there is one target job, which includes the TDDM.

The SDDM runs within a source job or activation group on the source server. It can handle multiple DDM conversations with one or more target servers at the same time. For the same source job or activation group, one SDDM handles all the remote file access requests. This is true regardless of how many target servers or remote files are involved. No separate job for the SDDM exists in the server.

If the source server DDM files involved all use the same remote location information to identify the target server, one TDDM job is created for each source server job that requests access to one or more files on the target server.

The following figure shows five programs accessing six DDM files. The numbers in the upper set of boxes representing DDM files correspond to the same numbers in the lower set of boxes representing the associated remote files. These DDM files are using four different remote location descriptions to access six different remote files, all on the same target server. Seven DDM conversations are needed to handle the processing. An explanation of the DDM conversations follows:

- PGM1 and PGM2 run in different source jobs and are using DDM files (2 and 3) that contain the same remote location information. A separate conversation is needed for each source job.
- PGM3 in source job 3 uses the two DDM files (5 and 6) that both use the same remote location information. They will share the same conversation and target job (5B).
- PGM4 and PGM5 run in different activation groups within source job 4. They are using two DDM files (5 and 6) that both use the same remote location information. A separate conversation is needed for each activation group.

In the following figure, jobs 1, 2, and 3 in System A each have a SDDM. Each activation group in job 4 has its own SDDM. Jobs 1B through 7B each have their own TDDM.

When the application program or the source job closes the DDM file on the source server, the DDM conversation and its associated target job ends, unless the following items are true:

- The value of the DDMCNV attribute of the Change Job (CHGJOB) command for the source job is *KEEP (the server default).
- Any locks established during the job by the Allocate Object (ALCOBJ) command still exist.

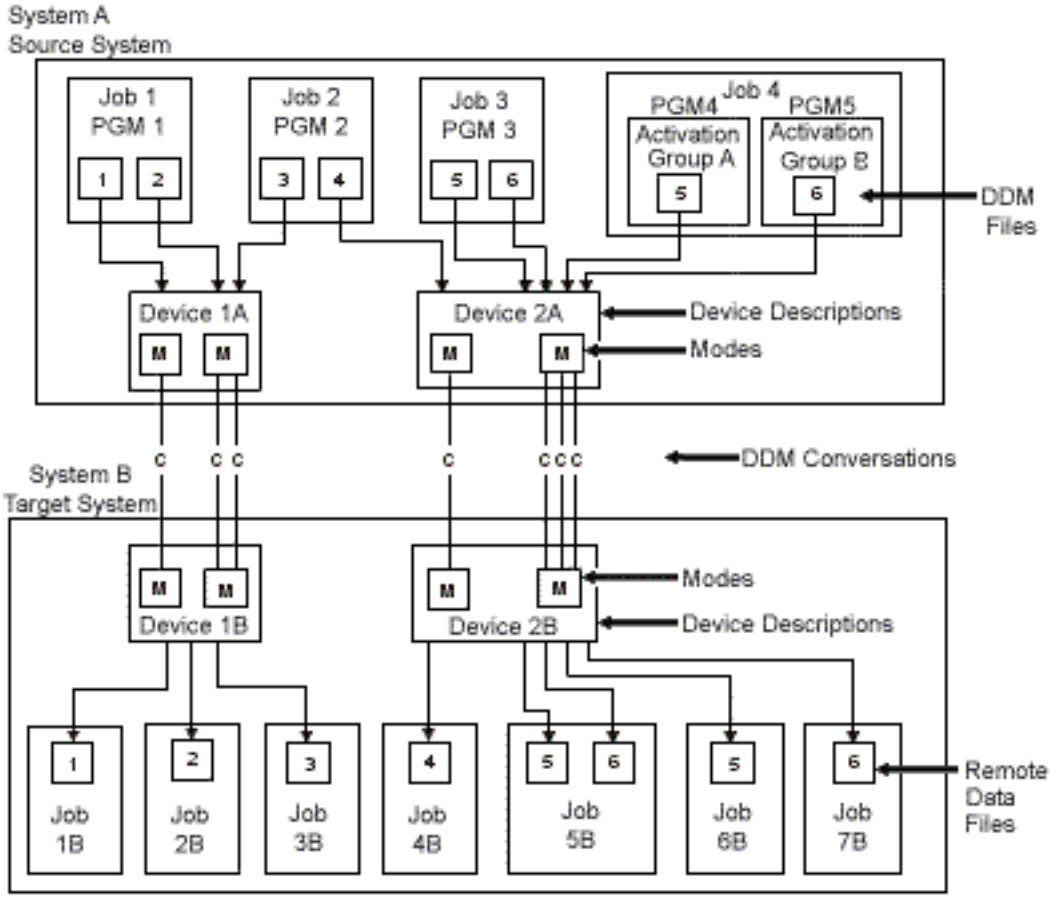


Figure 9. Relationships of DDM source and target jobs

The CHGJOB and ALCOBJ commands are described in topic Use CL and DDS with DDM. If DDMCNV(*KEEP) is specified, the DDM conversation remains active and waits for another DDM request to be started.

From a performance viewpoint, if the DDM conversation is likely to be used again, *KEEP is the value that should be used. This saves the time and resources used on the target server to start each TDDM and establish the conversation and job.

The following figure shows the relationship between the SDDM and two TDDMs on *different* target servers and the figure in Example: Access files on multiple servers with DDM topic shows the relationship between the SDDM and two TDDMs on *one* target server.

An iSeries server can be a source server and a target server at the same time, and two servers can be accessing files located on each other. In addition, an iSeries job can be a source job and a target job. A DDM file can refer to a remote file that is another DDM file.

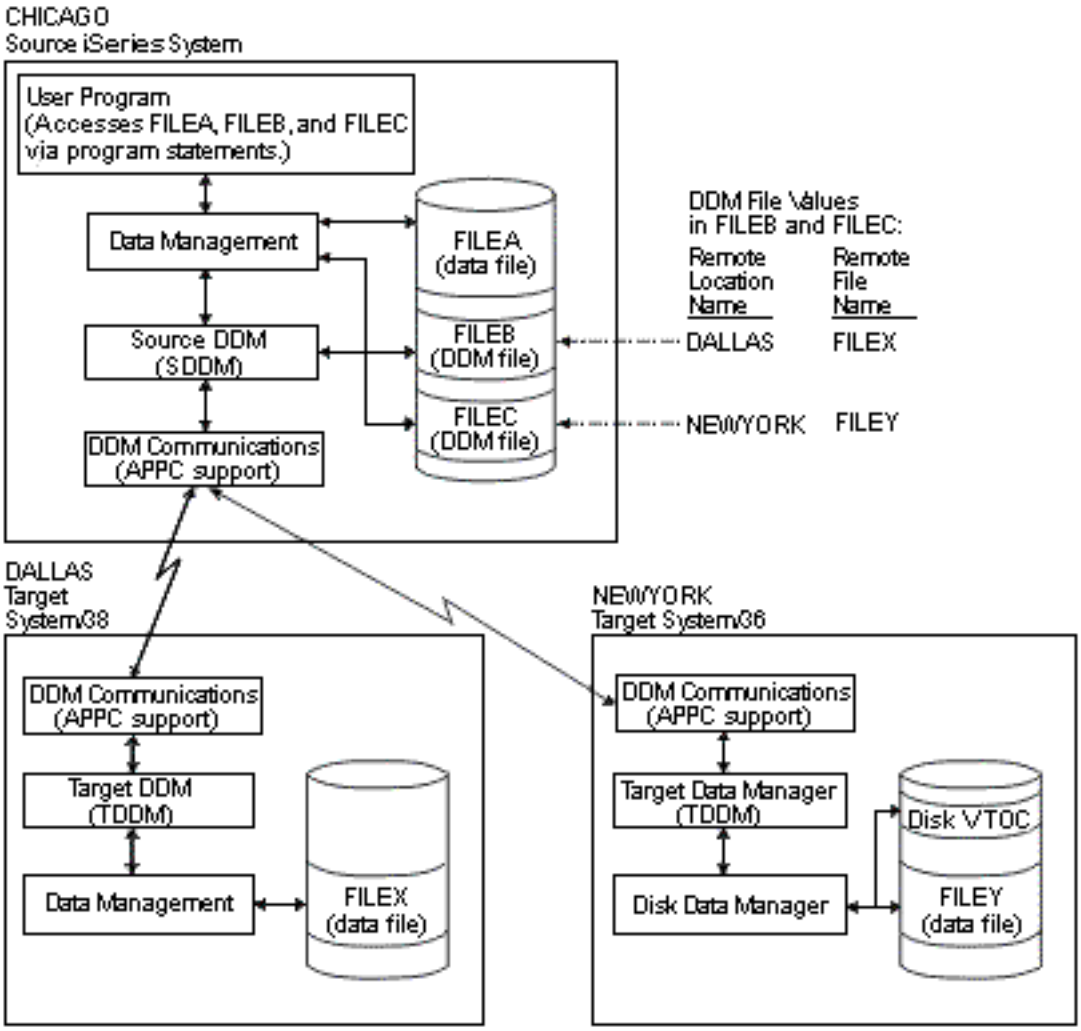


Figure 10. Example: Access multiple local and remote files. An iSeries server with communications links to a System/38 and to a System/36.

Related concepts

"Use CL and DDS with DDM" on page 66
This topic contains DDM-related information about specific iSeries control language (CL) commands, data description specifications (DDS) considerations, DDS keywords, and DDM user profile authority.

Related reference

"Example: Access files on multiple servers with DDM" on page 23
This topic contains a figure which shows the relationships among the source server, its DDM files, and two target servers.

"Additional considerations: SBMRMTCMD command" on page 72
This topic describes additional considerations for the SBMRMTCMD command.

Related information

ILE Concepts PDF

Examples: Access multiple remote files with DDM

These examples show a single application program using DDM to access multiple remote files.

The first example shows the remote files on different target servers, and the second shows them on the same target server.

Example: Access files on multiple servers with DDM

This topic contains a figure which shows the relationships among the source server, its DDM files, and two target servers.

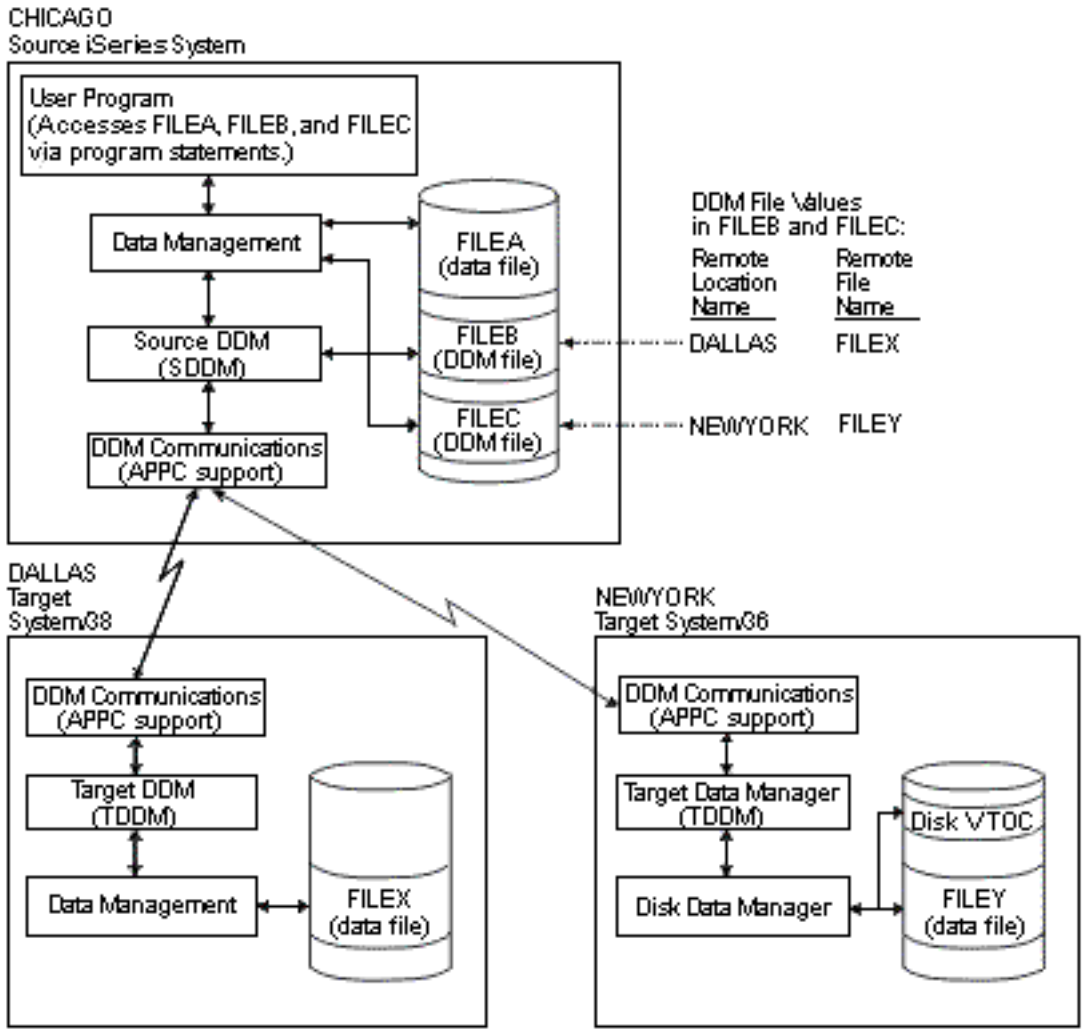


Figure 11. Example: Access multiple local and remote files. An iSeries server with communications links to a System/38 and to a System/36.

One target server is a System/38 and the other is a System/36. Each system has DDM installed.

The user program running on the source server is shown accessing three files: FILEA, FILEB, and FILEC. FILEA, located on the source server, is accessed using only local data management. On different target servers, DDM file FILEB corresponds to remote file FILEX and FILEC corresponds to remote file FILEY. When the program opens FILEB and FILEC, DDM allows the program to access the corresponding remote files as if they were on the source server. Only the person who defines the DDM files needs to know where each file is located or what the file's name is on the remote server.

Example: Process multiple requests for remote files with DDM

This example shows how multiple programs access multiple files on the same target server.

This example shows a System/36 target server. The SDDM is shown handling requests for two files from two programs in different jobs, and two TDDMs are handling the requests on the target server (one TDDM for each requesting program). Notice that, although program B is accessing two files on the target server, only one TDDM is created if all the associated DDM files specify the same remote location information to identify the target server.

Notice that both programs A and B are sharing FILEA. However, because these programs are shown to be in separate jobs, they *cannot* share the same open data path (ODP) to FILEA. If they were in the same job, programs A and B can share both the ODP on the source server *and* the remote file. When multiple programs within the same job are accessing a remote file at the same time (by using one TDDM for each program), the rules for file sharing are the same for remote files as for local files. These rules are based on how the SHARE parameter is specified on the Create DDM File (CRTDDMF), the Override with Database File (OVRDBF), and the Change DDM File (CHGDDMF) commands.

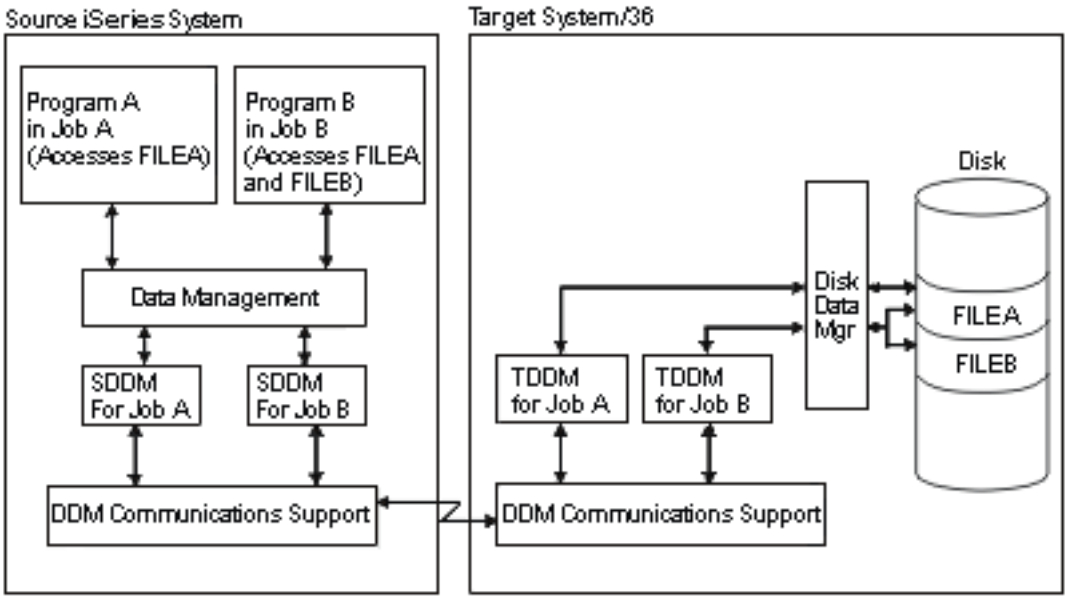


Figure 12. Example: Process multiple program and file requests

Use language, utility, and application support for DDM

This topic describes the language, utility, and application program support that is provided on the iSeries server for DDM.

This topic indicates which languages, utilities, and application programs support DDM, and provides any DDM-specific information needed to properly access remote files. Language-specific information concerning access to Customer Information Control System for Virtual Storage (CICS®) files is in topic iSeries server-to-CICS considerations with DDM.

Related concepts

"Source server actions dependent on type of target server" on page 17
If the target server is not another iSeries server or System/38, only the DDM architecture commands defined in Level 2.0 and earlier of the DDM architecture are used.

"iSeries server-to-CICS considerations with DDM" on page 194
This topic describes programming considerations for accessing CICS remote files with i5/OS DDM.

Programming language considerations for DDM

DDM is supported by these iSeries languages.

- ILE RPG
- ILE COBOL
- iSeries BASIC (interpretive and compiled forms)
- iSeries PL/I
- Control Language (CL) (interactive and compiled forms)
- ILE C

Note: iSeries Pascal does not support DDM.

DDM considerations for all languages

DDM files can be used as data files or source files by high-level language (HLL) programs.

However, for CL, data description specifications (DDS), and BASIC, if a DDM file is to be used as a source file, the target server must be an iSeries server or a System/38, and the file referred to by the DDM file must be defined on the target iSeries server or System/38 as a source file. That is, the remote file must have been created either by the Create Source Physical File (CRTSRCPF) command or as FILETYPE(*SRC) by the Create Physical File (CRTPF) command. These restrictions are not enforced by the ILE RPG, ILE COBOL, and ILE C compilers, which allow source files to be used from both iSeries and non-iSeries target servers.

If a source file *member* name is specified when the target server is not an iSeries server or a System/38, all the HLL compilers end compilation if the name of the source member specified on the SRCMBR parameter is different from the name of the DDM file specified on the SRCFILE parameter.

If programs that accessed local files are to access remote files, certain restrictions might require that a program be changed and recompiled. And, if the target server is not an iSeries server or a System/38, externally described data must, in some cases, reside on the local (source) server. All of these restrictions are described under the topic Program modification requirements for DDM.

If the target system is not an iSeries server or a System/38, the number of records returned in the open feedback might not be valid.

If you do not specify a library name for the SRCFILE parameter, the first file found in the user's library list with the same name as the file you specified for the SRCFILE parameter is used as the source file.

Related reference

"Program modification requirements for DDM" on page 45
Remote files can be accessed by iSeries application programs written in the HLL and control language.

HLL program input and output operations with i5/OS DDM:

The high-level language operations are supported by DDM for keyed or nonkeyed operations.

See the information in the following tables.

Table 1. High-level language operations supported by DDM for keyed or nonkeyed operations

	High-level languages			
i5/OS database operation	**ILE RPG programming language**	**ILE COBOL programming language**	**BASIC**	**PL/I**
Open file Query file	OPEN	OPEN	OPEN	OPEN

Table 1. High-level language operations supported by DDM for keyed or nonkeyed operations (continued)

	High-level languages			
i5/OS database operation	**ILE RPG programming language**	**ILE COBOL programming language**	**BASIC**	**PL/I**
Read (keyed access)	CHAIN (key)	READ INVALID KEY	READ KEY	READ EQUAL
Read first/last[1]	*LOVAL *HIVAL	READ FIRST LAST	READ FIRST LAST	READ FIRST LAST
Read next	READ READE[2]	READ <NEXT> AT END	READ	READ NEXT
Read previous	READP	READ PRIOR AT END	READ PRIOR	READ PRV
Read next or previous[3] Next equal Previous equal Next unique Previous unique			READ = =, PRIOR	READ NXTEQL PRVEQL NXTUNQ PRVUNQ
Read (relative to start)[4]	CHAIN (rrn)	READ RELATIVE KEY	READ REC=	READ KEY
Release record lock	EXCPT or next I/O op	(next I/O op)	(next I/O op)	(next I/O op)
Force end of data	FEOD			
Position file[5]	SETGT SETLL	START KEY GREATER KEY NOT LESS KEY EQUAL	RESTORE	
Update record	UPDAT	REWRITE[6]	REWRITE	REWRITE
Write record	WRITE/ EXCPT	WRITE[6]	WRITE	WRITE
Delete record	DELET	DELETE[6]	DELETE	DELETE
Close file	CLOSE	CLOSE	CLOSE	CLOSE

[1] For the ILE RPG language, if the keyed access path of a file specifies DESCENDING, then *LOVAL gets the last record in the file and *HIVAL gets the first record in the file.

[2] For duplicate keyed files, the ILE RPG language performs a READ NEXT operation and compares the key of the returned record to determine if the record qualifies. If so, the record is returned to the program; if not, an end-of-file indication is returned.

[3] If the remote file is on a non-iSeries server, these operations cannot be performed using DDM.

[4] An iSeries application program can open a *keyed* access open data path to a file and then access its records using both keyed and *relative record* access methods. Although DDM supports the *combined-access* access method, a target server (such as System/36) might not. In this case, the i5/OS can do relative record accessing of a keyed file on a non-iSeries target server if the target server supports the *combined-by-record-number* access method and if the DDM file specifies that method. The combined-by-record-number access method is specified on an iSeries server as ACCMTH(*ARRIVAL *BOTH) on the Create DDM File (CRTDDMF) command. If these values are not specified for the DDM file and the target server does not support the combined-access access method, relative record operations to a keyed file are rejected.

[5] Positioning operations (SET*xx* in the ILE RPG language, or START in the ILE COBOL language) do not return the record data to the application program. These operations also cause the file to be opened for random processing.

[6] ILE COBOL operations that change indexed or relative files can lock the record before the operation to make the record eligible.

Table 2. High-level language operations supported by DDM for keyed or nonkeyed operations

	High-level languages	
i5/OS database operation	**CL**	**ILE C programming language**
Open file	OPNDBF	FOPEN, FREOPEN
Query file	OPNQRYF	

Table 2. High-level language operations supported by DDM for keyed or nonkeyed operations (continued)

i5/OS database operation	High-level languages	
	CL	**ILE C programming language**
Read (keyed access)		
Read first/last		
Read next	RCVF	FREAD, FGETC
Read previous		
Read next or previous: Next equal Previous equal Next unique Previous unique		
Read (relative to start)		
Release record lock		(next I/O op)
Force end of data		FFLUSH
Position file	POSDBF	FSEEK, FSETPOS
Update record		FWRITE, FPUTC, FFLUSH
Write record		FWRITE, FPUTC, FFLUSH
Delete record		
Close file	CLOF	FCLOSE

Commitment control support for DDM

iSeries applications can commit or roll back transactions on remote iSeries servers.

However, DDM does not support the iSeries journaling commands (CRTJRN, CRTJRNRCV, and STRJRNPF). Before running applications, a user must create a journal on the target iSeries servers for recoverable resources to be used under commitment control, start journaling the physical files that are to be opened under commitment control, and issue the Start Commitment Control (STRCMTCTL) command on the source server. The STRCMTCTL command does not Support the Notify Object (NTFOBJ) command for DDM files. Another way to setup journaling on the remote server is to use the SBMRMTCMD DDM support to submit the journal commands to the target server to journal the remote files.

For DDM conversations to use two-phase commitment control, the DDM conversations need to be protected. For DDM conversations to be protected, the appropriate DDM file must have been created with the protected conversation (PTCCNV) parameter set to *YES.

Related concepts

Commitment control

Related reference

"DDM architecture-related restrictions" on page 45
The items listed in this topic are DDM architecture-related restrictions. Therefore, application programs that use these items might have to be changed and recompiled before they can access remote files.

Use DDM files with commitment control:

DDM files can be opened under commitment control.

However, the following restrictions should be considered when working with these DDM files:

- If more than one DDM file (with PTCCNV(*NO)) is opened under commitment control, the following items must be the same for each file:

  - Remote location name
  - Local location name
  - Device
  - Mode
  - Remote network ID
  - Transaction program name (TPN)
  - User ID
  - Activation group number
  - Open scope

  The exception to this rule is when all of the DDM files opened under commitment control are scoped to the job level. In this case, the activation group numbers are ignored and do not need to match.
- If a DDM file and a remote SQL object (Distributed Relational Database Architecture, DRDA) are running under commitment control (with PTCCNV(*NO)), the following items must be the same for the file and object:
  - Remote location name
  - Local location name
  - Device
  - Mode
  - Remote network ID
  - TPN
  - User ID
  - Activation group number
  - Open scope
- If the DDM file (with PTCCNV(*YES)) is being opened for output, update, or delete (not opened for input only), then there cannot be any one-phase DDM or DRDA conversations active.
- If a DDM with PTCCNV of *YES is being used, it must point to a target iSeries server that supports two-phase commitment control protocols.
- DDM files (with PTCCNV(*NO)) and local database files cannot be opened under commitment control at the same time within the same activation group.
- DDM files (with PTCCNV(*NO)) and local database files cannot be opened under commitment control at the same time within the same job if commitment control is scoped to the job level.
- To open a DDM file under commitment control and scope it to the job level, you must have specified CMTSCOPE(*JOB) on the Start Commitment Control (STRCMTCTL) command.
- You cannot use the Submit Remote Command (SBMRMTCMD) command to call programs that expect commitment control to be scoped to the job level. Because commitment control is always scoped to the activation group level in DDM target jobs, the program fails.
- The SBMRMTCMD command should not be used to start or end commitment control.
- The target server specified from the iSeries server working under commitment control must be another iSeries server.

Note: If the communications line fails during a COMMIT operation, the source and target servers will do a ROLLBACK operation. However, the target server might successfully complete the COMMIT operation before the line fails, but the source server will always do a ROLLBACK operation.

Table 3. High-level language commit and rollback commands

Operation	ILE RPG programming language	ILE COBOL programming language	PL/I	CL	ILE C programming language
Commit changes in transaction	COMMIT	COMMIT	PLICOMMIT	COMMIT	_Rcommit
Cancel entire transaction	ROLBK	ROLLBACK	PLIROLLBACK	ROLLBACK	_Rrollback

ILE RPG considerations for DDM

ILE RPG programs and automatic report programs can both refer to DDM files. Generally, DDM file names can be specified in ILE RPG programming language anywhere a database file name can be specified, for both iSeries and non-iSeries target servers.

- DDM file names can be specified on the Create RPG Program (CRTRPGPGM) and Create Auto Report Program (CRTRPTPGM) commands:
  - To access remote files containing source statements, on an iSeries server or a non-iSeries server, a DDM file name can be specified on the SRCFILE parameter, and a member name can be specified on the SRCMBR parameter.
    - For iSeries or System/38 target servers, a remote iSeries or System/38 source file (and, optionally, member) can be accessed in the same manner as a local source file and member.
    - For non-iSeries target servers, a remote source file can be accessed if both the PGM and SRCMBR parameter defaults are used on either command. Or, if a member name is specified, it must be the same as the DDM file name specified on the SRCFILE parameter. (The same is true for member names specified either on the /COPY statement of the input specifications used to create an automatic report program or as used by the compiler to include source specifications.)
  - To place the compiler listing in a database file on a target server, a DDM file name can be specified on the PRTFILE parameter of either command.
- A DDM file name and member name can be specified on the OUTFILE and OUTMBR parameters of the CRTRPTPGM command, but before the output produced by the command can be stored in the remote file referred to by the DDM file, the remote file must already exist. Also, as with local files, the record format of the remote file must match the required OUTFILE parameter format. Generally, this means that the target server must be an iSeries server or a System/38.

When an ILE RPG program opens a DDM file on the source server, the following types of I/O operations can be performed on the remote file at the target server, for both iSeries and non-iSeries targets: CHAIN, CLOSE, DELET, EXCPT, FEOD, OPEN, READ, READE, READP, SETGT, SETLL, UPDAT, and WRITE.

Other considerations are:

- If the DDM file is declared in the program to be externally described, the ILE RPG compiler copies the external descriptions of the remote file referred to into the program at compile time. However, if the remote file is not on an iSeries server or a System/38, the field declares for the record descriptions do not have meaningful names. Instead, all of the field names are declared as F*nnnnn* and the key fields are declared as K*nnnnn*.

  A recommended method for describing remote files, when the target is not an iSeries server or a System/38, is to have the data description specifications (DDS) on the local server and enter a Create Physical File (CRTPF) command or a Create Logical File (CRTLF) command on the local server. Compile the program using the local file name. Ensure that the remote system's file has the corresponding field types and field lengths.

  To access the remote file, use the Override with Database File (OVRDBF) command preceding the program, for example:

  ```
  OVRDBF FILE(PGMFIL) TOFILE(DDMFIL) LVLCHK(*NO)
  ```

- A DDM file is also valid as the file specified in the ILE RPG program that will be used implicitly in the ILE RPG logic cycle.
- A record format name, if used, must match the DDM file name when the target server is not an iSeries server or a System/38.
- An ADDROUT file created on a System/36 cannot be used on an iSeries server. iSeries System/36-Compatible RPG II uses 3-byte ADDROUT files, and ILE RPG programming language on an iSeries server and System/38 uses 4-byte ADDROUT files.

ILE COBOL considerations for DDM

ILE COBOL programs can refer to DDM files. Generally, DDM file names can be specified in ILE COBOL programming language anywhere a database file name can be specified, for both iSeries and non-iSeries target servers.

- DDM file names can be specified on the Create COBOL Program (CRTCBLPGM) command:
  - To access remote files containing source statements, on an iSeries server or a non-iSeries server, a DDM file name can be specified on the SRCFILE parameter, and a member name can be specified on the SRCMBR parameter.
    - For iSeries or System/38 target servers, a remote iSeries or System/38 source file (and, optionally, member) can be accessed in the same manner as a local source file and member.
    - For non-iSeries target servers, a remote source file can be accessed if both the PGM and SRCMBR parameter defaults are used on the CRTCBLPGM command. Or, if a member name is specified, it must be the same as the DDM file name specified on the SRCFILE parameter.
  - To place the compiler listing in a database file on a target server, a DDM file name can be specified on the PRTFILE parameter of the CRTCBLPGM command.
- DDM file names can be specified as the input and output files for the ILE COBOL SORT and MERGE operation. (The work file for this operation cannot be a DDM file.)
- A DDM file can be used in the ILE COBOL COPY statement when the DDS option on that statement is used to copy one or all of the externally described record formats from the remote file referred to by the DDM file into the program being compiled. If this is done when the remote file is not on an iSeries server or a System/38, the field declares for the record descriptions will not have meaningful names. Instead, all of the field names are declared as F*nnnnn* and the key fields are declared as K*nnnnn*.

  A recommended method for describing remote files, when the target is not an iSeries server or a System/38, is to have the data description specifications (DDS) on the local server and enter a Create Physical File (CRTPF) command or a Create Logical File (CRTLF) command on the local server. Compile the program using the local file name. Ensure that the remote server's file has the corresponding field types and field lengths.

  To access the remote file, use the Override with Database File (OVRDBF) command preceding the program, for example:

  ```
  OVRDBF FILE(PGMFIL) TOFILE(DDMFIL) LVLCHK(*NO)
  ```
- DDM file names can be specified on a COPY statement:
  - If you do not specify the library name with the file name, the first file found with that file name in the user's library list is used as the include file.
  - If the target server is not an iSeries server or a System/38, a DDM file name can be specified as the include file on a COPY statement, but the member name must be the same as the DDM file name.
- If the target server is a System/36, ILE COBOL programming language cannot be used to open a DDM file for output if the associated remote file has logical files built over it. For System/36 files with logical files, the open operation (open output) will fail because ILE COBOL programming language attempts to clear the file before using it.

When an ILE COBOL program opens a DDM file on the source server, the following statements can be used to perform I/O operations on the remote file at the target server, for both iSeries and non-iSeries targets: CLOSE, DELETE, OPEN, READ, REWRITE, START, and WRITE.

Direct file support with ILE COBOL:

An iSeries server does not support direct files as one of its file types. However, an ILE COBOL program on iSeries server can specify that a file be accessed as a *direct* file.

An iSeries server normally creates direct files as *sequential* files. An ILE COBOL program on an iSeries server defines a file as a direct file by specifying RELATIVE on the SELECT statement. If the program is to open the file for output only (by specifying OUTPUT on the OPEN statement), the file must be created with deleted records and contain no active records. This is also the file's condition when a non-iSeries source server (such as System/36) uses DDM to create or clear the direct file on an iSeries server, assuming that the file is created as described in the following paragraphs.

An iSeries server and System/38 support sequential and keyed file types. DDM recognizes sequential, keyed, and direct file types. For a non-iSeries server to create a direct file on an iSeries server using DDM, the DDM architecture command Create Direct File (CRTDIRF) is used.

When the CRTDIRF architecture command is issued from a non-iSeries server to create the file, the file is created as a physical file and is designated as a direct file so that, for subsequent direct file access by non-iSeries source servers, it will be identifiable to the other server as a direct file. If the file is not created in this way, an iSeries server cannot later determine whether the file is a direct file or a sequential file, again, because an iSeries server does not have direct files as one of its file types.

Therefore, if an ILE COBOL program on a server other than an iSeries server or a System/38 needs to access an iSeries or a System/38 file in a direct mode (that is, by relative record number) for output, the file must have been created by the CRTDIRF architecture command.

To support direct files on an iSeries server for output only, the ILE COBOL OPEN statement clears and prepares a member of a file being opened. Therefore, existing iSeries or System/38 files can be accessed by using DDM files by ILE COBOL programs on other iSeries servers or System/38s. For non-iSeries target servers, relative files opened for output must be defined as direct files or an error occurs.

In summary:

- If a file is created on the local iSeries server as a direct file by a program or user from a *non*-iSeries server, the file can be accessed as a direct file by an ILE COBOL program from a remote non-iSeries source server.
- If a file is created on the local iSeries server by a program or user on the *same* iSeries server, it cannot be accessed as a direct file by a non-iSeries server because the iSeries target server cannot determine, in this case, whether the file is a direct or sequential file.
- Any files created by a remote server can be used locally.

BASIC considerations for DDM

Compiled BASIC programs and interpretive BASIC statements can refer to DDM files. In addition, DDM file names can be specified on the Create BASIC Program (CRTBASPGM), Start BASIC (STRBAS), and Execute BASIC Procedure (EXCBASPRC) commands.

- A DDM file name can be specified on the SRCFILE parameter, and a member name can be specified on the SRCMBR parameter of the CRTBASPGM, STRBAS, and EXCBASPRC commands, but only if the remote source file (and member) is on an iSeries server or a System/38. If one of these commands refers to remote files on non-iSeries or non-System/38 target servers, the operation fails.
- A DDM file can be used as the source file for the following BASIC commands in the BASIC session: FREE, LOAD, MERGE, PROC, REPLACE, SAVE, SRCFILE, and SUBPROC. It can also be used in the CHAIN BASIC statement.
- A DDM file name can be specified in the DECLARE FILE statement. The remote file that the DDM file refers to is used to bring in the field definitions for an externally described file. If this is done and the

remote file is not on an iSeries server or a System/38, the field declares for the record descriptions will not have meaningful names. Instead, all of the field names are declared as F*nnnnn* and the key fields are declared as K*nnnnn*.

A recommended method for describing remote files, when the target is not an iSeries server or a System/38, is to have the data description specifications (DDS) on the local server and enter a Create Physical File (CRTPF) command or a Create Logical File (CRTLF) command on the local server. Compile the program using the local file name. Ensure that the remote server's file has the corresponding field types and field lengths.

To access the remote file, use the Override with Database File (OVRDBF) command preceding the program, for example:

```
OVRDBF FILE(PGMFIL) TOFILE(DDMFIL) LVLCHK(*NO)
```

- A DDM file can be specified as the file used in the LISTFMT and LISTFMTP BASIC commands. These commands extract the file descriptions of the referred to remote file to list any fields used in the program.

When BASIC is used to open a DDM file on the source server, the following statements can be used to perform I/O operations on the remote file at the target server, for both iSeries and non-iSeries targets: CLOSE, DELETE, INPUT, LINPUT, OPEN, READ, REREAD, RESTORE, REWRITE, and WRITE statements for processing record files, and GET and PUT statements for processing remote stream files.

PL/I considerations for DDM

Compiled programs can refer to DDM files. In addition, DDM file names can be specified on the Create PL/I Program (CRTPLIPGM) command.

- A DDM file name can be specified on the SRCFILE parameter, and a member name can be specified on the SRCMBR parameter, but only if the remote source file is on an iSeries server or a System/38. The same is true for specifying DDM file and member names on the %INCLUDE source directive statement. If the remote file referred to by the DDM file is not on an iSeries server or a System/38, an error occurs if a DDM file name is specified on the CRTPLIPGM command or %INCLUDE statement.
- When a DDM file is accessed as the source file for a program, the margins used in the compilation of the source are the default values of 2 and 72. No other margin values can be specified.
- If a %INCLUDE DDS directive statement specifies the name of a DDM file, the record descriptions of the remote file are included in the compiled program. However, if the remote file is not on an iSeries server or a System/38, the field declares for the record descriptions do not have meaningful names. Instead, all of the field names are declared as F*nnnnn* and the key fields are declared as K*nnnnn*.

A DDM file can be used to refer to remote record files or remote stream files. When a program opens a DDM file on the source server, the following types of statements can be used to perform I/O operations on the remote file at the target server, for both iSeries and non-iSeries targets: OPEN, CLOSE, READ, WRITE, REWRITE, and DELETE statements for processing record files, and GET and PUT statements for processing stream files.

Another consideration is if the target server is not an iSeries server or a System/38, the POSITION parameter on a keyed READ statement to read from a remote file does not work if a value of NXTEQL, PRVEQL, NXTUNQ, or PRVUNQ is specified for the parameter. (The values of NEXT, PREVIOUS, FIRST, and LAST do work.) All the values are valid if the target system is an iSeries server or a System/38.

CL command considerations for DDM

Both compiled control language (CL) programs and interactively entered CL commands can refer to DDM files.

Generally, DDM file names can be specified in CL commands anywhere a database file name can be specified for both iSeries and non-iSeries target servers. But there are some limitations.

Listed here are some examples of where DDM file names can be specified:

- DDM file names can be specified on many of the database file-related commands, such as the copy, display, and override file commands.
- DDM file names can be specified on the create file commands to access remote *source* files, but only if the target server is an iSeries server or a System/38. A DDM file name can be specified on the SRCFILE parameter, and a member name can be specified on the SRCMBR parameter. If the remote source file referred to by the DDM file is not on an iSeries server or a System/38, an error occurs. The considerations for remote iSeries or System/38 source members are the same as for local source members.
- DDM file names can be specified on the FILE parameter of the Declare File (DCLF) command.

When a DDM file name is specified, some commands act on files on the source server, some act on target files, and some parameter values allow you to specify either a source or target file.

Related concepts

"Use CL and DDS with DDM" on page 66
This topic contains DDM-related information about specific iSeries control language (CL) commands, data description specifications (DDS) considerations, DDS keywords, and DDM user profile authority.

"Operating considerations for DDM" on page 111
This topic provides task-oriented information and examples that describe various aspects of DDM operation considerations.

Related reference

"DDM-related CL command summary charts" on page 104
This topic shows summary charts containing most of the control language (CL) commands used with DDM.

ILE C considerations for DDM

ILE C programs can refer to DDM files.

Generally, DDM file names can be specified in ILE C programming language anywhere a database file name can be specified, for both iSeries and non-iSeries target servers.

Specify DDM file names on the Create C Program (CRTCPGM) command to do the following items:

- Access remote files on an iSeries or non-iSeries server that contains source statements. To do this, specify a DDM file name on the SRCFILE parameter, and a member name on the SRCMBR parameter.

  Notes:

  1. For iSeries or System/38 target systems, you access a remote iSeries or System/38 source file (or member) in the same manner as a local source file and member.
  2. For non-iSeries target servers, access a remote source file by using the same file name for the SRCMBR and the SRCFILE parameters.
- Place the compiler listing in a database file on a target server. To do this, specify a DDM file name on the PRTFILE parameter of the CRTCPGM command.

When using ILE C programming language, consider the following items:

- If the target system is not an iSeries server or a System/38, you can specify a DDM file name as the include file on the #INCLUDE source directive statement, but the member name must be the same as the DDM file name.
- ILE C programming language only supports sequential I/O operations.
- Although ILE C programming language does not directly support keyed files, key exceptions might occur if you are using a keyed file.

Utility considerations for DDM

These iSeries utilities support DDM for accessing remote files.

Notes:

1. The following utilities do *not* support DDM: iSeries Query, source entry utility (SEU), screen design aid (SDA), and advanced printer function utility.
2. Except when the System/38-compatible database tools or DFU/400 is being used, DDM does not support displaying lists of members in remote files. However, if the target server is an iSeries server or a System/38, display station pass-through can be used to perform this function.
3. The SQL/400® licensed program and query management, part of the i5/OS licensed program, do not support DDM. However, both support the Distributed Relational Database Architecture (DRDA) in a distributed network.

System/38-compatible database tools

The topic collection describes the System/38-compatible data file utility (DFU/38) and the System/38-compatible query utility (Query/38).

System/38-compatible data file utility (DFU/38):

DFU/38 data entry applications can be created and used with DDM to work with remote files in the same manner as with local files.

If a remote file is on an iSeries server or System/38, most DFU/38 functions are performed with the remote file as though it is a local file. When creating or changing a DFU/38 application and the remote file is a logical file, the following consideration applies: either DDM files referring to each remote based-on file must exist on the source server, and the DDM file and library names must match those of the remote based-on files; or, alternatively, physical files with the same file and library names and the same record formats as the remote based-on files must exist on the source server. Because only the record formats are needed from the physical files, they need not contain data. Using this alternative, if the record formats of the remote based-on files are changed, the record formats on the source server must also be changed so that the record formats match.

However, DFU/38 does *not* support non-iSeries or non-System/38 target systems. If you attempt to use DFU/38 with non-iSeries or non-System/38 remote files, you might experience processing problems when trying to change or delete records in such a file. Although an iSeries server does not prevent any user from creating and using such an application, the default field descriptions created on the source iSeries server for the non-iSeries or non-System/38 remote file would probably be too general to be useful. (These files appear to be physical files with one member, whose member name is the same as the file name. The file has one record format and within that format: one field for the entire record, if it is a nonkeyed file; two fields for keyed files, one for the key and one for the remainder of the record; or more than two fields for keyed files with separate key fields.)

All the DFU/38 commands can be used in applications that access local files or DDM files. And, wherever a local database file name can be specified on any of the DFU command parameters, a DDM file can also be specified, as long as any other limitations are met.

A DDM file name can be specified in the SRCFILE parameter of the Create DFU Application (CRTDFUAPP) or Retrieve DFU Source (RTVDFUSRC) command, but only if the target server is an iSeries server or a System/38 and if the target file is a source physical file.

System/38-compatible query utility (Query/38):

The System/38-compatible query utility (Query/38) can be used with DDM to create and use interactive or batch query applications.

If the target server is an iSeries server or a System/38, most of these functions can be performed as though the remote file is a local file. When creating or changing a Query/38 application and the remote file is a logical file, the following consideration applies: either DDM files referring to each remote based-on file must exist on the source server, and the DDM file and library names must match those of the remote based-on files; or, alternatively, physical files with the same file and library names and the same record formats as the remote based-on files must exist on the source server. Because only the record formats are needed from the physical files, they need not contain data. Using this alternative, if the record formats of the remote based-on files are changed, the record formats on the source server must also be changed so that the record formats match.

If the target system is not an iSeries server or a System/38, you should refer to a local file for the format and fields that describe the data in the remote file, and then use the Override Database File (OVRDBF) command to override the local file with a DDM file when the Query/38 application is run. The local file used to create (or re-create) the query must have the same record format name as the source description of the non-iSeries or non-System/38 target file. The default record format name is the name of the source DDM file.

Although Query/38 can create an application that uses a file on a non-iSeries or non-System/38 system, the default field descriptions created on the source iSeries server for the non-iSeries remote file probably would be too general to be useful. (These files appear to be physical files with one member, whose member name is the same as the file name. The file has one record format and within that format: one field for the entire record, if it is a nonkeyed file; two fields for keyed files, one for the key and one for the remainder of the record; or more than two fields for keyed files with separate key fields.)

Related reference

"i5/OS database query" on page 38
The database interactive query function, provided by the i5/OS licensed program, supports DDM files.

"Non-iSeries or non-System/38 Query/38 example"
This example shows how to create a local file and use it to define the data that is to be queried in a non-iSeries or non-System/38 remote file.

"OPNQRYF (Open Query File) command" on page 94
You can query remote files using the Open Query File (OPNQRYF) command, but only if the remote files are on a target iSeries server or a target System/38.

Non-iSeries or non-System/38 Query/38 example:

This example shows how to create a local file and use it to define the data that is to be queried in a non-iSeries or non-System/38 remote file.

Assume that a DDM file named RMTS36FILE exists on your iSeries server and it refers to a remote System/36 file that you want to query. You can perform the following steps to: determine the attributes of the remote System/36 file; locally create a physical file that has the attributes of the remote file; and define, create, and run the Query/38 against the remote file.

1. Use the Display File Field Description (DSPFFD) command and specify SYSTEM(*RMT) to display the attributes of the remote file associated with the RMTS36FILE DDM file.

   ```
   DSPFFD  FILE(RMTS36FILE) SYSTEM(*RMT)
   ```

   In this example, the displayed results would show that the remote file's record length is 80 characters, its record format name is RMTS36FILE, and it has two fields: K00001, with 12 characters (starting in position 1), and F00001, with 68 characters (starting in position 13). The K in field K00001 indicates it is the key field for this format.
2. Using the DDS and the preceding information before defining your Query/38 application, create a local physical file and call it LCLS36FILE. The DDS might look something like this:

```
A          R RMTS36FILE
A            CUSNO           6A
A            BILLCODE        6A
A            ADDR1          15A
A            ADDR2          15A
A            ADDR3          15A
A            ZIP             5A
A            AMTOWE          7S 2
A            OUTBAL          7S 2
A            MISC            4A
A          K CUSNO
A          K BILLCODE
```

Three main rules must be followed when defining the local file:

- The record format name must be the same as the record format name displayed by the Display File Field Description (DSPFFD) command.
- Key integrity must be maintained. In this case, the key must be 12 characters long, and must start at the beginning of the file in position 1.
- The total record length must be the same as the record length displayed by the DSPFFD command.

3. Define your Query/38 application using the local file created in step 2. Because the remote file is a non-iSeries file, OPTIMIZE(*NO) should be specified on the query command.
4. Before your Query/38 application is run, issue the following Override Database File (OVRDBF) command:

   ```
   OVRDBF  FILE(LCLS36FILE) TOFILE(RMTS36FILE)
   ```

   When the Query/38 application is run, this command overrides the local file you created with the DDM file that is associated with the desired target file.
5. Run your Query/38 application using the Query Data (QRYDTA) command. The net effect is that a query of the remote file is done using the local file description.

Related reference

"System/38-compatible query utility (Query/38)" on page 34
The System/38-compatible query utility (Query/38) can be used with DDM to create and use interactive or batch query applications.

"Query/38 optimization for DDM" on page 37
Query/38 has an optimization function, but because it causes i5/OS database query to be used, the feature cannot be used when the query is performed against a remote file that is not on an iSeries server or a System/38.

Query/38 output considerations for DDM:

Query/38 output to an existing non-iSeries or a non-System/38 target file is possible, but only under specific circumstances.

Query/38 allows output to any local or remote file only if the file is sequential and if its field attributes match those attributes required by the Query/38 application. If both conditions are not met, Query/38 rejects the specified output file before the Query/38 application runs.

Because the source server description of a non-iSeries or a non-System/38 target file is very general, its field attributes probably do not match the attributes required by the Query/38 application. Therefore, in most cases, Query/38 rejects that file if it is specified for output. It works, however, if the Query/38 output consists of one alphanumeric field only, and if the record length of the target file is large enough to hold this field.

Query/38 command considerations for DDM:

All the Query/38 commands can be used in applications that access local files or DDM files. And, wherever a local database file name can be specified on any of the Query/38 command parameters, a DDM file can also be specified, as long as any other limitations are met.

Note: If a Query/38 command uses a DDM file associated with a remote file on a non-iSeries or a non-System/38 target server, either the DDM file should specify LVLCHK(*NO) or an OVRDBF command should be used to override that parameter with *NO. This is recommended to avoid level-checking problems with the target file.

A DDM file name can be specified in the SRCFILE parameter of the Create Query Application (CRTQRYAPP) or Retrieve Query Source (RTVQRYSRC) command, but only if the target server is an iSeries server or a System/38 and if the target file is a source physical file.

Query/38 optimization for DDM:

Query/38 has an optimization function, but because it causes i5/OS database query to be used, the feature cannot be used when the query is performed against a remote file that is not on an iSeries server or a System/38.

Because i5/OS database query does not exist on non-iSeries servers or non-System/38s, the optimization function cannot be used by the source iSeries server when performing a query against a non-iSeries or a non-System/38 remote file.

Therefore, when a Query/38 application is being created or changed that accesses a remote file on a non-iSeries server or a non-System/38, the OPTIMIZE parameter on the Create Query Application (CRTQRYAPP), Create Query Definition (CRTQRYDEF), or Change Query Definition (CHGQRYDEF) command must be changed to *NO. Specifying OPTIMIZE(*NO) forces Query/38 to read the file sequentially, which can be done with non-iSeries target files. If the default of *YES is used, an error occurs when the Query/38 application is run.

Similarly, if the Design Query Application (DSNQRYAPP) command is used to create and run queries that are to be performed on a non-iSeries target file, the *Optimize Query* prompt on the Application Creation display must be changed from Y to N.

Related reference

"Non-iSeries or non-System/38 Query/38 example" on page 35
This example shows how to create a local file and use it to define the data that is to be queried in a non-iSeries or non-System/38 remote file.

"i5/OS database query" on page 38
The database interactive query function, provided by the i5/OS licensed program, supports DDM files.

Existing Query/38 application considerations for DDM:

You should keep these considerations in mind for existing Query/38 applications.

Existing Query/38 applications, if they are to query remote files, must be re-created in all cases, even if the target server is an iSeries server or a System/38. If the target server is an iSeries server or a System/38, the re-created application that uses a DDM file is defined and run as if the remote file is a local file. The optimization feature can be used to get the records from the target iSeries server or the target System/38.

Data file utility for iSeries server

Data file utility (DFU) data entry applications can be created and started with DDM to work with remote files in the same manner as with local files. Most DFU functions are performed with the remote file as though it were a local file.

When you create or change a DFU function of Application Development Tools and the remote file is an iSeries or System/38 logical file, the following consideration applies: either DDM files referring to each remote based-on file must exist on the source server, and the DDM file and library names must match those of the remote based-on files; or, alternatively, physical files with the same file and library names and the same record formats as the remote based-on files must exist on the source server. Because only the record formats are needed from the physical files, they need not contain data. Using this alternative, if the record formats of the remote based-on files are changed, the record formats on the source server must also be changed so that the record formats match. Similar considerations apply when the remote file is a System/36 logical file.

DFU supports iSeries server, System/38, and System/36 remote files. However, DFU does not prevent you from using non-iSeries, non-System/38, or non-System/36 remote files and you might experience problems when using such files.

Non-iSeries or System/36 files are program-described files. DFU allows you to use either a local or remote file containing ILE RPG file and input specifications to define these data files.

i5/OS database query

The database interactive query function, provided by the i5/OS licensed program, supports DDM files.

This support is used by iSeries Access Family and System/38-compatible query utility if OPTIMIZE(*YES) is specified. You can query remote files using the Open Query File (OPNQRYF) command, but only if the remote files are on a target iSeries server or a target System/38.

The query utility on the System/38 can be used to query remote files that are not from an iSeries server.

Related reference

"System/38-compatible query utility (Query/38)" on page 34
The System/38-compatible query utility (Query/38) can be used with DDM to create and use interactive or batch query applications.

"Query/38 optimization for DDM" on page 37
Query/38 has an optimization function, but because it causes i5/OS database query to be used, the feature cannot be used when the query is performed against a remote file that is not on an iSeries server or a System/38.

"OPNQRYF (Open Query File) command" on page 94
You can query remote files using the Open Query File (OPNQRYF) command, but only if the remote files are on a target iSeries server or a target System/38.

Multiple remote files:

Database query allows accessing of either multiple local files or multiple remote files (by using DDM files) at the same time, but not both.

If all the files are remote, they must all reside on the same target server. Also, the DDM files that refer to the remote files must all specify the same remote location information. If this restriction is not met, an error message is displayed to the user of iSeries Access Family or to the user of the Open Query File (OPNQRYF) command who requested the query.

Sort utility

The sort utility supports remote file processing with DDM anywhere that it supports local file processing for both iSeries and non-iSeries target servers.

Generally, on the Format Data (FMTDTA) command, DDM file names can be specified anywhere a database file name can be specified.

- A DDM file name can be specified on the SRCFILE parameter, and a member name can be specified on the SRCMBR parameter, for iSeries or System/38 target systems. If the remote file referred to by the DDM file is not on an iSeries server or a System/38, a member name cannot be specified.
- DDM file names can also be specified on the INFILE parameter (to access a remote file as the input file for conversion) or on the OUTFILE parameter (to access a remote file as the output file of the conversion). Both parameters cannot specify DDM file names at the same time.

iSeries Access Family considerations for DDM

The iSeries Access Family supports DDM for accessing remote files, with limitations.

Note: iSeries Business Graphics Utility does not support DDM.

The transfer function in iSeries Access Family can be used with DDM to transfer data between a personal computer attached to a local iSeries server and another remote server.

When the transfer function is being used, the remote system must be an iSeries system or a System/38. The iSeries Access Family copy commands, Copy to PC Document (CPYTOPCD) and Copy from PC Document (CPYFRMPCD), can be used to copy data on a host server or between host servers.

The figure here shows a personal computer attached to the local iSeries server. The iSeries Access Family user can access data on remote servers through a DDM file defined on the local iSeries server. The iSeries server with the personal computer attached can only be the source server.

- The **iSeries Access Family transfer function** can be used by a personal computer user to transfer data from a remote file to the personal computer, or to transfer data from the personal computer to a remote file. Only a personal computer user can start the requests, not an iSeries user.

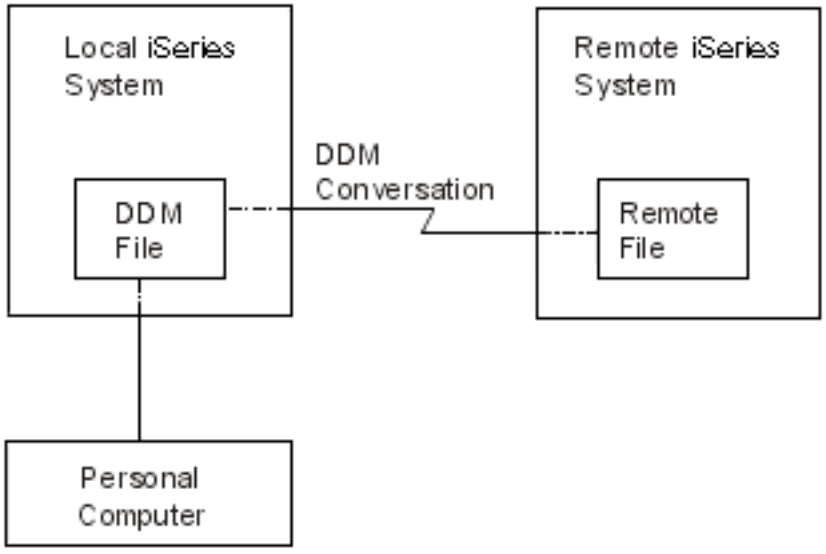


Figure 13. Use DDM with iSeries Access Family

- The **iSeries Access Family copy commands** can be used with DDM to copy data from a personal computer document located on the local iSeries server to a database file on the remote iSeries server, or to copy data to a personal computer document on the local iSeries server from a database file on the remote iSeries server.

Note: For iSeries Access Family, database query allows accessing of multiple remote files (by using DDM files) at the same time.

iSeries Access Family transfer function considerations

A personal computer user can use the transfer function in iSeries Access Family and the DDM support on the local iSeries server to which the personal computer is attached either to transfer data *from* the personal computer to a remote file, or to transfer data from a remote file *to* the personal computer.

The remote file must be on an iSeries server or a System/38.

When DDM is used to transfer files or data from a remote server *to* an attached personal computer, the DDM files (that refer to remote files) on the local iSeries server cannot be joined with local files to transfer data to the personal computer. (That is, data from files on both the remote and local servers cannot be joined.) However, a DDM file can specify a remote file that is a logical join file built over multiple physical files. DDM files that refer to the same target server and use the same remote location information can be joined.

A transfer request that requires group processing does not work if the local server is a System/38 and the remote server is an iSeries server, or if the local server is an iSeries server and the remote server is a System/38.

When DDM is used to transfer a file or data *from* an attached personal computer to a remote server, a remote file cannot be created on the target server. The remote file must already exist before the data from the personal computer can be transferred. However, because the target must be an iSeries server or a System/38, a new member can be added in the remote file before personal computer data is transferred to that file member.

iSeries Access Family copy command considerations

The iSeries CL command Copy from Personal Computer Document (CPYFRMPCD) used in iSeries Access Family can be used to copy data *from* a document located either on an iSeries server *to* a database file member located on the same iSeries server or on a remote iSeries server using DDM.

The CL command Copy to Personal Computer Document (CPYTOPCD) can be used to copy data *from* a database file member on a local iSeries server or a remote iSeries server (using DDM) *to* a document on the local iSeries server. The remote file can be on a target iSeries server or a non-iSeries server. To use these commands, specify the name of a DDM file on the:

- TOFILE parameter in the Copy from PC Document (CPYFRMPCD) command, to copy a personal computer document to an iSeries physical file.
- FROMFILE parameter in the Copy to PC Document (CPYTOPCD) command, to copy a member from an iSeries database file to a personal computer document in a folder.

The following restrictions apply to the CL copy commands for iSeries Access Family:

- For the CPYFRMPCD command, a remote file cannot be created on the target server (whether it is an iSeries server or a non-iSeries server). The remote file must already exist before the personal computer document data can be copied to it. However, if the target is an iSeries server or a System/38, a new member can be created for the remote file before the personal computer document data is copied to that file member.
- The CPYFRMPCD and CPYTOPCD commands are iSeries CL commands and cannot be entered at the DOS prompt from the personal computer.

For more information on the CPYTOPCD and the CPYFRMPCD commands, see the online help information.

Hierarchical file system API support for DDM

The hierarchical file system (HFS) APIs and the functions that they support are part of the i5/OS operating system.

The APIs provide applications with a single, consistent interface to all the hierarchical file systems available on your iSeries server. They automatically support the document library services (DLS) file system and can support user-written file systems also.

DDM can be registered under HFS as one of the user-written file systems. DDM, however, only supports the copy stream file (QHFCPYSF) HFS API. To register DDM under HFS, you must run the following command on your iSeries source system, `CALL QTSREGFS`. If no errors occur, DDM is successfully registered with HFS.

Calling DDM using the HFS QHFCPYSF API causes one of two DDM-architected commands to be generated, the LODSTRF (load stream file) or ULDSTRF (unload stream file) command. Both of these DDM commands are part of the stream file DDM model (STRFIL). If the DDM target server you are working with does not support the STRFIL DDM model, then errors will occur when trying to use this support. DDM uses documents and folders (DLS) on the server to copy stream file data either to (ULDSTRF case) or from (LODSTRF case).

To use the DDM HFS copy stream file support, note the following items:

- Both the source and destination file path names must begin with the string '/QDDM/' to indicate to HFS that DDM is the file system that will handle the copy stream file function.
- The copy information HFS parameter is ignored by DDM, but you still must pass a valid HFS value.
- Either the source or destination file path name parameter must be the name of a DDM file, but not both. The DDM file used must point to a target server that supports the STRFIL DDM file model and the remote file name value must end with the string ' FMS' if the DDM file points to another iSeries server.
- The other source or destination file path name parameter that is not a DDM file, must be the name of an existing DLS object (document in a folder) and the name must be followed by the string ' FMS'.
- The maximum source or target path name length supported by DDM is 63 characters. The 63 characters do not include the '/QDDM/' or the ' FMS' possible appendages.
- In the LODSTRF case (source file path name is a local DLS object and target file path name is a DDM file), the local DLS document is read starting at offset zero through the end of the file. Whether the destination file (pointed to by the DDM file) exists or not is dependent on the target server's stream file support.
- In the ULDSTRF case (source file path name is a DDM file and destination file path name is a local DLS object), the local or target DLS document must exist on the iSeries and will have its contents cleared and then written to starting at offset zero.

Here is a copy stream file example that will generate a LODSTRF DDM command to a remote server:

```
CRTDDMF FILE(DDMLIB/DDMFILE) +
RMTFILE(*NONSTD 'TARGET/SYSTEM/
SYNTAX/PATHNAME FMS') RMTLOCNAME(RMTSYSNM)
```

In this example, the local DLS object is 'PATH1/PATH2/FOLDER1/DOC1'.

You would call QHFCPYSF with the following parameter list:

```
1   Source file path name = '/QDDM/PATH1/PATH2/FOLDER1/DOC1 FMS'
2   Source file path name length = 34
3   Copy information = valid HFS value that is ignored by DDM
4   Target file path name = '/QDDM/DDMLIB/DDMFILE'
5   Target file path name length = 20
```

Just reverse the source and destination file path names and lengths to generate an ULDSTRF DDM command.

The example program in the following example calls DDM HFS API:

Note: By using the code examples, you agree to the terms of the "Code license and disclaimer information" on page 209.

```
/*********************************************************************/
/*********************************************************************/
/*  FUNCTION: This program copies a stream file using the QHFCPYSF  */
/*            HFS API.                                               */
/*                                                                   */
/*  LANGUAGE: PL/I                                                   */
/*                                                                   */
/*  APIs USED: QHFCPYSF                                              */
/*                                                                   */
/*********************************************************************/
/*********************************************************************/
TRANSFER: PROCEDURE(SRCFIL,TRGFIL) OPTIONS(MAIN);

/* parameter declarations                                           */
DCL SRCFIL CHARACTER (73);
DCL TRGFIL CHARACTER (73);

/* API entry declarations                                           */
/*                                                                  */
/* The last parameter, the error code, is declared as FIXED BIN(31) */
/* for the API. This always has a value of zero, specifying that    */
/* exceptions should be returned.                                   */
DCL QHFCPYSF ENTRY(CHAR(73),FIXED BIN(31),CHAR(6),CHAR(73),
                   FIXED BIN(31),FIXED BIN(31))
                   OPTIONS(ASSEMBLER);

/*********************************************************************/
/*  Parameters for QHFCPYSF                                          */
/*********************************************************************/
DCL srclen FIXED BIN(31);
DCL trglen FIXED BIN(31);
DCL cpyinfo CHAR(6);
DCL error_code FIXED BIN(31);

/*********************************************************************/
/*  Mainline routine                                                 */
/*********************************************************************/

srclen = INDEX(SRCFIL,'  ') - 1;
trglen = INDEX(TRGFIL,'  ') - 1;
cpyinfo = '1     ';
error_code = 0;
/* Copy the stream file                                             */
Call QHFCPYSF(SRCFIL,srclen,cpyinfo,TRGFIL,trglen,
              error_code);

END TRANSFER;
```

Figure 14. Program example

Sample command source that can be used with the preceding program:

```
CMD
     PARM    KWD(SRCFIL) TYPE(*CHAR) LEN(73) +
               PROMPT('SOURCE FILE NAME')
     PARM    KWD(TRGFIL) TYPE(*CHAR) LEN(73) +
               PROMPT('TARGET FILE NAME')
```

Related concepts

Application programming interfaces

Related reference

Hierarchical File System APIs

"LODSTRF (Load Stream File) Level 2.0" on page 180
This command sends a whole stream file from the source server to the target server. This command is sent by a source iSeries server when using the copy stream file HPS API.

"ULDSTRF (Unload Stream File) Level 2.0" on page 192
This command sends a document from the target to the source. This command is sent by a source iSeries server when using the Copy Stream File (QHFCPYSF) HFS API.

Prepare to use DDM

There are several requirements that must be met for DDM to be used properly.

Notes:

- Before determining which files should be accessed using DDM, review Performance considerations for DDM.
- Programming requirements and considerations for control language (CL) commands and data description specifications (DDS) are covered in Using CL and DDS with DDM and Operating considerations for DDM.

Related concepts

"Performance considerations for DDM" on page 134
These topics provide information to help you improve performance when using DDM and also provide some information about when to use something other than DDM to accomplish some functions.

"Use CL and DDS with DDM" on page 66
This topic contains DDM-related information about specific iSeries control language (CL) commands, data description specifications (DDS) considerations, DDS keywords, and DDM user profile authority.

"Operating considerations for DDM" on page 111
This topic provides task-oriented information and examples that describe various aspects of DDM operation considerations.

Communications requirements for DDM in an APPC network

Each iSeries server in a DDM network that is not using OptiConnect must have these communications requirements.

- The APPC/APPN support or the iSeries Access Family licensed program installed and configured on the server.
- At least one Systems Network Architecture (SNA) communications line connection that uses synchronous data link communications (SDLC), token-ring network, Ethernet, or X.25 protocol.

The number of sessions that can be used for DDM conversations is not limited by DDM. The maximum is determined in the same manner as for any other APPC-related communications. For parallel sessions, the session maximum is specified in the mode. For single session devices, the session maximum is always one.

iSeries servers in a DDM network that use OptiConnect must have the OptiConnect software and hardware installed. OptiConnect replaces the need for SNA communications line connections.

Related concepts

APPC, APPN, and HPR

OptiConnect

Communications Configuration PDF

Connecting to System i: iSeries Access for Windows

Configure a communications network in a TCP/IP network

This topic provides a high-level overview of the steps you take to set up a TCP/IP network.

1. Identify your iSeries server to the local network (the network that your iSeries server is directly connected to).
   a. Determine if a line description already exists.
   b. If a line description does not already exist, create one.
   c. Define a TCP/IP interface to give your iSeries server an IP address.
2. Define a TCP/IP route. This allows your iSeries server to communicate with servers on remote TCP/IP networks (networks that your iSeries server is not directly connected to).
3. Identify the names of the servers in your network.
   a. Build a local host table.
   b. Identify a remote name server.
4. Start TCP/IP.
5. Verify that TCP/IP works.

Security requirements for DDM

You can prevent intentional and unintentional access to the data resources of a system by the DDM user.

Access to data in the DDM environment can be limited—or prevented altogether—by a server-level network attribute, the DDMACC parameter on the Change Network Attributes (CHGNETA) command on the server. This attribute allows the server (as a target server) to prevent all remote access; or it allows the server to control file access by using standard authority to files and, further, by using an optional user exit program to restrict the types of operations allowed on the files for particular users.

To provide adequate security, you might need to set up additional user profiles on the target server, one for each source server user who can have access to one or more target server files. Or, a default user profile should be provided for multiple source server users. The default user profile is determined by the communications entry used in the subserver in which the target jobs are run.

For user profiles (or their equivalent) on non-iSeries target servers, refer to that server's documentation.

Related concepts

"Security" on page 48
This topic describes how iSeries security relates to DDM, and how it can limit access to the data resources of a target server by source server programs and users.

DDM file requirements

Before remote files can be accessed by an iSeries server, DDM files must be created on the source server.

At the time a DDM file is used, the device (remote location name) and mode (APPC session characteristics) specified in the DDM file must also exist on the server if APPN is not used. If APPN is used, then the device does not need to exist on the server. However, the server identified by the remote location name must exist within the APPN network. The APPN parameter on the Create Controller Description (APPC) (CRTCTLAPPC) and the Create Controller Description (SNA Host) (CRTCTLHOST) commands controls whether APPN is used.

Related concepts

"Use CL and DDS with DDM" on page 66
This topic contains DDM-related information about specific iSeries control language (CL) commands, data description specifications (DDS) considerations, DDS keywords, and DDM user profile authority.

Program modification requirements for DDM

Remote files can be accessed by iSeries application programs written in the HLL and control language.

In most cases, these applications can access both local or remote files without the programs being changed. However, some considerations and restrictions might require the programs to be changed and recompiled. These are grouped in three categories:

- iSeries functions that are not supported by the DDM architecture, but for which a System/38 extension to the architecture might exist. These functions can be used only when the source and target servers are System/38s or iSeries servers.
- Restrictions and considerations that apply when the *source* or *target* server is an iSeries server.
- Restrictions and considerations that apply to all target servers (iSeries servers and non-iSeries servers). User programs accessing local files should program for abnormal conditions such as *No record found*, *End of file*, and *Record lock time-out on read for update*. These conditions can also occur when a remote file is being accessed using DDM. In addition, the use of DDM exposes the program to communication line failures while sending disk I/O operations.

  When a communications failure occurs, the server sends an appropriate message to the job, which is returned to the application program as a generic file error. Each high-level language provides unique user syntax capabilities for user-controlled handling or default processing of exceptional results of a disk operation. Some languages might permit the user to retrieve the job message identification (ID) that would specifically indicate a DDM communications failure. Refer to the appropriate language manual for specific capabilities.

  For secondary SDLC lines, it is recommended that the INACTTMR parameter of the Create Line Description (SDLC) (CRTLINSDLC) command be set on the source and target servers to detect the stopping of polling by the primary server. This prevents the possibility of a DDM read-for-update record lock lasting indefinitely due to a communications failure on the primary server.

DDM architecture-related restrictions

The items listed in this topic are DDM architecture-related restrictions. Therefore, application programs that use these items might have to be changed and recompiled before they can access remote files.

- The DDM architecture does not support iSeries multi-format logical files. However, because multi-format logical files are supported as an System/38 extension to the DDM architecture, they can be used with DDM, but only if the source and target servers are iSeries servers or System/38s.
- Externally described data (using data description specifications [DDS] on an iSeries server) is not supported by the DDM architecture. However, DDS can still be used, especially if both systems are iSeries servers or System/38s. If the target server is an iSeries server or a System/38, most of the DDS support can be used as though the remote file is a local file.
- To access folder management services objects, the source server must support Level 2.0 or Level 3.0 of the DDM architecture for stream files and the stream access method. The following restrictions for the byte stream model apply:
  - WAIT time is not supported by the folder management services on the Lock Data Stream (LCKSTR) command. The user must handle the waiting function on the source server.
  - The Copy File (CPYFIL) command used to copy a document on an iSeries server is supported with the restrictions. Only the header information is copied; no data is copied.
  - The DELDRCOP (DRCALL) parameter is not supported on the Delete Directory (DELDRC) command.
- Personal computer generic names are not allowed when performing operations on data management objects such as files, libraries, or members. However, generic names are allowed when performing operations on folder management services objects such as documents and folders. Generic names are supported where the personal computer supports the operation and in the manner that the personal computer supports the operation. For example, generic names are not supported for folders using the rename and delete commands because the personal computer does not support them.

Related reference

"Commitment control support for DDM" on page 27
iSeries applications can commit or roll back transactions on remote iSeries servers.

"Data description specifications considerations for DDM" on page 108
Data description specifications (DDS), which is used to externally describe the fields and record formats, can also be used with DDM to describe the file and record formats of a remote file.

"DDM commands and parameters" on page 165
This topic classifies DDM commands and parameters.

iSeries source and target restrictions and considerations for DDM

When the *source* server is an iSeries server, iSeries database functions can be used on remote files. There are some restrictions, however.

The restrictions are:

- A source iSeries server can create files on a System/38, but the DDM architecture file models are used. As a result, no multiformat logical or join logical files can be created on a non-iSeries target server, including a System/38.
- Save or restore operations do not save or restore the data on a target server; only the DDM file object can be saved or restored locally.
- Operations that delay for a time period (that is, that wait for a file or record) are determined by the time values specified on the target server. (These values are specified by the WAITFILE and WAITRCD parameters on various CL commands.) This can result in increased delay times when DDM is used to access files or records remotely.
- Query requests (OPNQRYF) to a System/38 cannot use group selection and join processing.
- When running System/36 applications to or from an iSeries server, these applications might result in time-outs while waiting for a resource to become available. When running System/36 applications to or from another System/36, the application waits indefinitely for the resource to become available.

  For both source and target DDM jobs, due to the way DDM sends APPC operations, it is possible for the DDM job on the secondary side of the APPC conversation to wait indefinitely after a line failure or other failures at the remote server.

  Consider the following suggestions to avoid indefinite waits:

  – If the remote server supports record lock time-outs, ensure reasonable time values are specified. For example, on a target iSeries server or System/38 database file, do not use maximum values for CRTPF ... WAITRCD.

    WAITRCD addresses read-for-update operations, but does not apply to other file operations, such as read only, add, and so on.
  – When using an SDLC secondary line, use a time value for the line inactivity timer (INACTTMR). Do *not* use the *NOMAX value.
  – Provide the person responsible for server operation with the associated line, controller, and device names (or a list of DDM jobs that might run). If a DDM job then appears to be waiting indefinitely, this person can display the job information to determine if the job is waiting indefinitely by reviewing the job's processing unit time use (by using the Display Job (DSPJOB) command to display the active run attributes).

When the *target* server is an iSeries server, iSeries database functions can be used to access remote files, with the following restrictions:

- The physical files that the logical files or join logical files are based on must exist on the same iSeries server.
- A logical file on a source iSeries server cannot share the access path of a remote file (on any target server).
- Query requests (OPNQRYF), which require group selection and join processing from a System/38, do not work.

Related concepts

Communications Management PDF

Non-iSeries target restrictions and considerations for DDM

In addition to the restrictions that apply when the target server is an iSeries server, the restrictions in this topic also might apply when the target server is not an iSeries server or a System/38.

Whether they apply depends on what the target server supports. You should refer to that server's documentation for more information.

- Only field data types that are common to the source and target servers can normally be processed by HLL applications. Floating-point data is an example of a data type that might not be common. Records can be transmitted that contain floating-point data, but the representation of floating-point data sent between servers might differ.

  The packed signs sent between systems might differ; for example, one server might use a C and another server might use an F.

  Note: It is possible for you to write your application program so that it interprets the byte string for a record processed through a DDM file in any way that you want. However, whenever you do this, it is your responsibility to ensure that the data is handled correctly.
- Any operations that request a delay period before returning, such as for record lock wait times, might be rejected or changed to a zero wait time by the target server.
- Lock requests can be changed by the target server to a more restrictive lock. This might prevent some operations from occurring at the same time that can otherwise be performed on the local iSeries server. See "ALCOBJ (Allocate Object) command" on page 82 for more information.
- Some iSeries parameters are ignored or cause errors if they are used during remote file processing on non-iSeries target servers. Examples are the FRCRATIO and FMTSLR parameters on some of the file commands. For more information, see OVRDBF (Override with Database File) command and see Copy commands with DDM.
- Member names are not supported in the DDM architecture. When the target server is not an iSeries server or a System/38, CL commands that have a MBR parameter, such as the Clear Physical File Member (CLRPFM) command, must be changed if the parameter specifies a member name that is different than the file name. If the member name is different, an error occurs if the command is used for a non-iSeries remote file. For some commands, MBR(*FIRST) or MBR(*LAST) is also valid. See Member-related commands with DDM for a list of all the CL commands related to file members, and for those that are not valid for accessing files on non-iSeries target servers.

  Note: MBR(*LAST) is not supported by System/38.
- If a parameter on a CL command requires the name of a source file, then the names of the DDM files that refer to non-iSeries target files cannot be specified. An iSeries server cannot determine whether a remote file on a non-iSeries target is in fact a source file. (See Source file commands for a list of all the CL commands related to source files.)
- Certain iSeries commands that are valid for iSeries or System/38 target servers are not valid for other targets. See DDM-related CL command lists for the lists of commands that are not supported when the target is not an iSeries server or a System/38.

Related reference

"OVRDBF (Override with Database File) command" on page 94
The Override with Database File (OVRDBF) command can be used with DDM to override (replace) a local database file named in the program with a DDM file; the DDM file causes the associated remote file to be used by the program instead of the local database file.

"Copy commands with DDM" on page 85
This topic describes the DDM implications of these CL commands.

"Member-related commands with DDM" on page 101
Database file operations that apply to a member can be used by DDM.

"Source file commands" on page 103
If the target server is an iSeries server or a System/38, these CL commands can support a DDM file as a source file (on the SRCFILE parameter).

Security

This topic describes how iSeries security relates to DDM, and how it can limit access to the data resources of a target server by source server programs and users.

The access to target iSeries data can be limited by using standard authority to files, standard authority to commands, and an optional user exit program in the DDM environment at the target server.

Security authentication is first performed when a remote user accesses the target iSeries. If the target iSeries is not able to authenticate the remote user, the conversation is rejected. Security authorization is performed when a remote user accesses an iSeries file. The remote user must be authorized to perform the operation (open, close, read, or write, for example), or the DDM request is rejected. Application programs on the iSeries server can be isolated from each other by object authorities. .

Related concepts

APPC, APPN, and HPR

"Performance considerations for DDM" on page 134
These topics provide information to help you improve performance when using DDM and also provide some information about when to use something other than DDM to accomplish some functions.

Related reference

"Security requirements for DDM" on page 44
You can prevent intentional and unintentional access to the data resources of a system by the DDM user.

Elements of distributed relational database security

A distributed relational database administrator needs to protect the resources of the application servers in the network without unnecessarily restricting access to data by *application requesters (ARs)* in the network.

An AR secures its objects and relational database to ensure only authorized users have access to distributed relational database programs. This is done using normal iSeries server object authorization to identify users and specify what each user (or group of users) is allowed to do with an object. Alternatively, authority to tables, views, and SQL packages can be granted or revoked using the SQL GRANT and REVOKE statements. Providing levels of authority to SQL objects on the AR helps ensure that only authorized users have access to an SQL application that accesses data on another system.

The level of system security in effect on the *application server (AS)* determines whether a request from an AR is accepted and whether the remote user is authorized to objects on the AS.

Some aspects of security planning for iSeries server in a distributed relational database network include:

- Object-related security to control user access to particular resources such as confidential tables, programs, and packages
- Location security that verifies the identity of other systems in the network
- User-related security to verify the identity and rights of users on the local system and remote systems
- Physical security such as locked doors or secured buildings that surround the systems, modems, communication lines and terminals that can be configured in the line description and used in the route selection process

Location, user-related, and object-related security are only possible if the system security level is set at level 20 or above.

For Advanced Program-to-Program Communication (APPC) conversations, when the system is using level 10 security, an iSeries server connects to the network as a nonsecure system. The server does not validate the identity of a remote system during session establishment and does not require conversation security on incoming program start requests. For level 10, security information configured for the APPC remote location is ignored and is not used during session or conversation establishment. If a user profile does not exist on the server, one is created.

When the system is using security level 20 or above, an iSeries server connects to the network as a secure system. The iSeries system can then provide conversation-level security functions and, in the case of APPC, session level security as well.

Having system security set at the same level across the systems in your network makes the task of security administration easier. An AS controls whether the session and conversation can be established by specifying what is expected from the AR to establish a session. For example, if the security level on the AR is set at 10 and the security level on the AS is above 10, the appropriate information might not be sent and the session might not be established without changing security elements on one of the systems.

Passwords for DRDA access

The most common method of authorizing a remote user for database access is by flowing a user ID and password at connection-time. One method an application programmer can use to do this is to code the USER/USING clause on an embedded SQL CONNECT statement. For example:

```
EXEC SQL CONNECT TO :locn USER :userid USING :pw
```

For Distributed Relational Database Architecture (DRDA) access to remote relational databases, once a conversation is established, you do not need to enter a password again. If you end a connection with either a RELEASE, DISCONNECT, or CONNECT statements when running with the remote unit of work (RUW) connection management method, your conversation with the first application server (AS) might or might not be dropped, depending on the kind of AS you are connected to and your application requester (AR) job attributes (for the specific rules, see Control DDM conversations). If the conversation to the first AS is not dropped, it remains unused while you are connected to the second AS. If you connect again to the first AS and the conversation is unused, the conversation becomes active again without you needing to enter your user ID and password. On this second use of the conversation, your password is also not validated again.

Elements of security in an APPC network

When Distributed Relational Database Architecture (DRDA) is used, the data resources of each server in the DRDA environment should be protected.

To protect data resources of each server in the DRDA environment, you can use three groups of security elements that are controlled by the following parameters:

- For system-related security or session, the *LOCPWD parameter* is used on each iSeries server to indicate the system validation password to be exchanged between the source and target systems when an Advanced Program-to-Program Communication (APPC) session is first established between them. Both systems must exchange the same password before the session is started. (On System/36, this password is called the location password.) In an APPC network, the LOCPWD parameter on the Create Device Description (APPC) (CRTDEVAPPC) command specifies this password. Devices are created automatically using APPN, and the location password on the remote location list specifies a password that is used by the two locations to verify identities. Use the Create Configuration List (CRTCFGL) command to create a remote location list of type (*APPNRMT).
- For user-related or location security, the *SECURELOC parameter* is used on each iSeries server to indicate whether it (as a target server) accepts incoming access requests that have their security already verified by the source server or whether it requires a user ID and encrypted password. In an APPC network, the SECURELOC parameter on the Create Device Description (APPC) (CRTDEVAPPC) command specifies whether the local server allows the remote server to verify security. Devices are

created automatically using APPN, and the secure-location on an APPN remote Configuration List is used to determine if the local server allows the remote server to verify user security information. The SECURELOC value can be specified differently for each remote location.

The SECURELOC parameter is used with the following security elements:

- The user ID sent by the source server, if allowed by this parameter
- The user ID and encrypted password, if allowed by this parameter
- The target server user profiles, including default user profiles

For more information, see the DDM source system security in an APPC network topic.

- For object-related security, the *DDMACC parameter* is used on the Change Network Attributes (CHGNETA) command to indicate whether the files on the iSeries server can be accessed at all by another server and, if so, at which level of security the incoming requests are to be checked. More information about this object-related parameter is provided in the topic DDM Network Attribute (DDMACC Parameter).
  - If *REJECT is specified on the DDMACC parameter, all DRDA requests received by the target iSeries server are rejected.
  - If *OBJAUT is specified on the DDMACC parameter, normal object-level security is used on the target server.
  - If the name of an optional, user-supplied user exit program (or access control program) is specified on the DDMACC parameter, an additional level of security is used. The user exit program can be used to control whether a given user of a specific source server can use a specific command to access (in some manner) a specific file on the target server. (See the topic DDM server access control exit program for additional security for details.)
  - When a file is created on the target server using DRDA, the library name specified contains the file. If no library name is specified on the DRDA request, the current library (*CURLIB) is used. The file authority defaults to allow only the user who created the file or the target server's security officer to access the file.

Most of the security controls for limiting remote file access are handled by the target server. Except for the user ID provided by the source server, all of these elements are specified and used on the target server. The source server, however, also limits access to target server files by controlling access to the DRDA file on the source server and by sending the user ID, when needed, to the target server.

Related reference

"DDM server access control exit program for additional security" on page 59
Customers who use menu-level security, which is accomplished by restricting the user's access to functions on the server, are likely to have a large number of public files. *Public files* are those files to which the public has some or all authority. A user exit program allows you to restrict each DDM user's access to public files and to private files.

APPN configuration lists:

In an APPC network, location passwords are specified for those pairs of locations that are going to have end-to-end sessions between them.

Location passwords need not be specified for those locations that are intermediate nodes.

The remote location list is created with the Create Configuration List (CRTCFGL) command, and it contains a list of all remote locations, their location password, and whether the remote location is secure. There is one system-wide remote location configuration list on an iSeries server. A central site iSeries server can create location lists for remote iSeries servers by sending them a control language (CL) program.

Changes can be made to a remote configuration list using the Change Configuration List (CHGCFGL) command, however, they do not take effect until all devices for that location are all in a varied off state.

When the Display Configuration List (DSPCFGL) command is used, there is no indication that a password exists. The Change Configuration List (CHGCFGL) command indicates a password exists by placing *PASSWORD in the field if a password has been entered. There is no way to display the password. If you have problems setting up location security you might have to enter the password again on both systems to be sure the passwords match.

Related concepts

APPC, APPN, and HPR

Conversation level security:

Systems Network Architecture (SNA) logical unit (LU) 6.2 architecture identifies three conversation security designations that various types of systems in an SNA network can use to provide consistent conversation security across a network of unlike systems.

The SNA security levels are:

SECURITY(NONE)
No user ID or password is sent to establish communications.

SECURITY(SAME)
Sign the user on to the remote server with the same user ID as the local server.

SECURITY(PGM)
Both a user ID and a password are sent for communications.

SECURITY(PROGRAM_STRONG)
Both a user ID and a password are sent for communications only if the password will not be sent unencrypted, otherwise an error is reported. This is not supported by DRDA on i5/OS.

While the iSeries server supports all four SNA levels of conversation security, DRDA uses only the first three. The target controls the SNA conversation levels used for the conversation.

For the SECURITY(NONE) level, the target does not expect a user ID or password. The conversation is allowed using a default user profile on the target. Whether a default user profile can be used for the conversation depends on the value specified on the DFTUSR parameter of the Add Communications Entry (ADDCMNE) command or the Change Communications Entry (CHGCMNE) command for a given subsystem. A value of *NONE for the DFTUSR parameter means the application server (AS) does not allow a conversation using a default user profile on the target. SECURITY (NONE) is sent when no password or user ID is supplied and the target has SECURELOC(*NO) specified.

For the SECURITY(SAME) level, the remote server's SECURELOC value controls what security information is sent, assuming the remote server is an iSeries. If the SECURELOC value is *NONE, no user ID or password is sent, as if SECURITY(NONE) had been requested; see the previous paragraph for how SECURITY(NONE) is handled. If the SECURELOC value is *YES, the name of the user profile is extracted and sent along with an indication that the password has already been verified by the local server. If the SECURELOC value is *VFYENCPWD, the user profile and its associated password are sent to the remote server after the password has been encrypted to keep its value secret, so the user must have the same user profile name and password on both servers to use DRDA.

Note: SECURELOC(*VFYENCPWD) is the most secure of these three options because the most information is verified by the remote server; however, it requires that users maintain the same passwords on multiple servers, which can be a problem if users change one server but do not update their other servers at the same time.

For the SECURITY(PGM) level, the target expects both a user ID and password from the source for the conversation. The password is validated when the conversation is established and is ignored for any following uses of that conversation.

DRDA application server security in an APPC network:

When the target server is an iSeries server, several elements are used together to determine whether a request to access a remote file is allowed or not.

User-related security elements

The user-related security elements include the SECURELOC parameter on the target server, the user ID sent by the source server (if allowed), the password for the user ID sent by the source server, and a user profile or default user profile on the target server.

Object-related security elements

The object-related security elements include the DDMACC parameter and, optionally, a user exit program supplied by the user to supplement normal object authority controls.

User-related elements of target security

A valid user profile must exist on the application server (AS) to process distributed relational database work. You can specify a default user profile for a subsystem that handles communications jobs on an iSeries server.

The name of the default user profile is specified on the DFTUSR parameter of the Add Communications Entry (ADDCMNE) command on the AS. The ADDCMNE command adds a communications entry to a subsystem description used for communications jobs.

If a default user profile is specified in a communications subsystem, whether the AS is a secure location or not determines if the default user profile is used for this request. The SECURELOC parameter on the Create Device Description (APPC) (CRTDEVAPPC) command, or the secure location designation on an APPN remote location list, specifies whether the AS is a secure location.

- If *YES is specified for SECURELOC or secure location on the AS, the AS considers the application requester (AR) a secure location. A user ID and an Already Verified indicator is expected from the AR with its request. If a user profile exists on the AS that matches the user ID sent by the requester, the request is allowed. If not, the request is rejected.
- If *NO is specified for the SECURELOC parameter on the AS, the AS does not consider the AR a secure location. Although the AR still sends a user ID, the AS does not use this for the request. Instead, a default user profile on the AS is used for the request, if one is available. If no default user profile exists on the AS, the request is rejected.
- If *VFYENCPWD is specified for SECURELOC on the AS, the AS considers the AR a secure location, but requires that the user ID and its password be sent (in encrypted form) to verify the identity of the current user. If the user profile exists on the AS that matches the user ID sent by the requester, and that requester has the same password on both systems, the request is allowed. Otherwise, the request is rejected.

The following table shows all of the possible combinations of the elements that control SNA SECURITY(PGM) on the iSeries server. A ″Y″ in any of the columns indicates that the element is present or the condition is met. An ″M″ in the PWD column indicates that the security manager retrieves the user's password and sends a protected (encrypted) password if password protection is active. If a protected password is not sent, no password is sent. A *protected password* is a character string that APPC substitutes for a user password when it starts a conversation. Protected passwords can be used only when the systems of both partners support password protection and when the password is created on a system that runs i5/OS or OS/400® Version 2 Release 2 or later.

Table 4. Remote access to a distributed relational database

Row	UID	PWD[1]	AVI	SEC(Y)	DFT	Valid	Access
1	Y	Y		Y	Y	Y	Use UID
2	Y	Y		Y	Y		Reject
3	Y	Y		Y		Y	Use UID
4	Y	Y		Y			Reject
5	Y	Y			Y	Y	Use UID
6	Y	Y			Y		Reject
7	Y	Y				Y	Use UID
8	Y	Y					Reject
9	Y		Y	Y	Y	Y	Use UID
10	Y		Y	Y	Y		Reject
11	Y		Y	Y		Y	Use UID
12	Y		Y	Y			Reject
13	Y	M[3]			Y	Y	Use DFT or UID[2]
14	Y	M[3]			Y		Use DFT or UID[2]
15	Y	M[3]				Y	Reject or UID[2]
16	Y	M[3]					Reject or UID[2]
17				Y	Y		Used DFT
18				Y			Reject
19					Y		Use DFT
20							Reject

Key:

UID User ID sent

PWD Password sent

AVI Already Verified Indicator set

SEC(Y) SECURELOC(YES) specified

DFT Default user ID specified in communication subsystem

Valid User ID and password are valid

Use UID
Connection made with supplied user ID

Use DFT
Connection made with default user ID

Reject Connection not made

Notes:

1. If password protection is active, a protected password is sent.
2. Use UID when password protection is active.
3. If password protection is active, the password for the user is retrieved by the security manager, and a protected password is sent; otherwise, no password is sent.

To avoid having to use default user profiles, create a user profile on the AS for every AR user that needs access to the distributed relational database objects. If you decide to use a default user profile, however, make sure that users are not allowed on the system without proper authorization. For example, the following command specifies the default user parameter as DFTUSER(QUSER); this allows the system to

accept job start requests without a user ID or password from a communications request. The communications job is signed on using the QUSER user profile.

```
ADDCMNE   SBSD(SAMPLE) DEV(*ALL) DFTUSER(QUSER)
```

Elements of security in a TCP/IP network

DDM and DRDA over native TCP/IP does not use i5/OS communications security services and concepts such as communications devices, modes, secure location attributes, and conversation security levels which are associated with Advanced Program-to-Program Communication (APPC). Therefore, security setup for TCP/IP is quite different.

Application requester security in a TCP/IP network:

Different connectivity scenarios call for using different levels of authentication. Therefore, an administrator can set the lowest security authentication method required by the application requester (AR) when connecting to an application server (AS) by setting the preferred authentication method field in each RDB directory entry.

The administrator might also allow the decision about authentication method to be negotiated with the server, by choosing to allow a lower security authentication method. In this case, the preferred authentication method is still attempted, but if the AS cannot accept the preferred method, a lower method can be used, depending on the server security setting and other factors such as the availability of cryptographic support. For example, if two systems are in a physically unprotected environment, the administrator might choose to require Kerberos authentication without allowing lower security authentication methods.

On the application requester (client) side, you can use one of the two methods to send a password along with the user ID on DRDA TCP/IP connect requests. If you do not use either of these methods, the CONNECT command can send only a user ID.

The first way to send a password is to use the USER/USING form of the SQL CONNECT statement, as in the following example from the interactive SQL environment:

```
CONNECT TO rdbname USER userid USING 'password'
```

In a program using embedded SQL, the values of the user ID and of the password can be contained in host variables in the USER/USING database.

In a program using CLI, the following example shows how the user ID and password are presented in host variables to the DRDA application requester (AR):

```
SQLConnect(hdbc,sysname,SQL_NTS,     /*do the connect to the application server */
     uid,SQL_NTS,pwd,SQL_NTS);
```

The second way to provide a password is to send a connect request over TCP/IP using a server authorization entry. A server authorization list is associated with every user profile on the system. By default, the list is empty; however, you can add entries by using the Add Server Authentication Entry (ADDSVRAUTE) command. When you attempt a DRDA connection over TCP/IP, the DB2® UDB for iSeries client (AR) checks the server authorization list for the user profile under which the client job is running. If it finds a match between the RDB name on the CONNECT statement and the SERVER name in an authorization entry (which must be in uppercase), the associated USRID parameter in the entry is used for the connection user ID. If a PASSWORD parameter is stored in the entry, that password is also sent on the connect request.

A server authorization entry can also be used to send a password over TCP/IP for a DDM file I/O operation. When you attempt a DDM connection over TCP/IP, DB2 UDB for iSeries checks the server authorization list for the user profile under which the client job is running. If it finds a match between either the RDB name (if RDB directory entries are used) or 'QDDMSERVER' and the SERVER name in an

authorization entry, the associated USRID parameter in the entry is used for the connection user ID. If a PASSWORD parameter is stored in the entry, that password is also sent on the connect request.

To store a password using the Add Server Authentication Entry (ADDSVRAUTE) command, you must set the QRETSVRSEC system value to '1'. By default, the value is '0'. Type the following command to change this value:

```
CHGSYSVAL QRETSVRSEC VALUE('1')
```

The following example shows the syntax of the Add Server Authentication Entry (ADDSVRAUTE) command when using an RDB directory entry:

```
ADDSVRAUTE USRPRF(user-profile) SERVER(rdbname) USRID(userid) PASSWORD(password)
```

The USRPRF parameter specifies the user profile under which the application requester job runs. What you put into the SERVER parameter is normally the name of the RDB to which you want to connect. The exception is if you are using DDM files which were not created to use the RDB directory. In that case, you should specify QDDMSERVER in the SERVER parameter. When you specify an RDB name, it must be in uppercase. The USRID parameter specifies the user profile under which the server job will run. The PASSWORD parameter specifies the password for the user profile.

If you omit the USRPRF parameter, it will default to the user profile under which the Add Server Authentication Entry (ADDSVRAUTE) command runs. If you omit the USRID parameter, it will default to the value of the USRPRF parameter. If you omit the PASSWORD parameter, or if you have the QRETSVRSEC value set to 0, no password will be stored in the entry and when a connect attempt is made using the entry, the security mechanism attempted will be user ID only.

You can use the Display Server Authentication Entries (DSPSVRAUTE) command to determine what authentication entries have been added to the server authentication list. The Retrieve Server Authentication Entries (QsyRetrieveServerEntries) (QSYRTVSE) API in a user-written program can also be used.

You can remove a server authorization entry by using the Remove Server Authentication Entry (RMVSVRAUTE) command. You can change a server authorization entry by using the Change Server Authentication Entry (CHGSVRAUTE) command

If a server authorization entry exists for a relational database (RDB), and the USER/USING form of the CONNECT statement is also used, the latter takes precedence.

Kerberos source configuration

Distributed Relational Database Architecture (DRDA) and distributed data management (DDM) can take advantage of Kerberos authentication if both systems are configured for Kerberos.

If a job's user profile has a valid ticket-granting ticket (TGT), the DRDA application requester (AR) uses this TGT to generate a service ticket and authenticate the user to the remote server. Having a valid TGT makes the need for a server authentication entry unnecessary, because no password is directly needed in that case. However, if the job's user profile does not have a valid TGT, the user ID and password can be retrieved from the server authentication entry to generate the necessary TGT and service ticket.

When using Kerberos, the remote location (RMTLOCNAME) in the RDB directory entry must be entered as the remote host name. IP addresses will not work for Kerberos authentication.

In cases where the Kerberos realm name differs from the DNS suffix name, it must be mapped to the correct realm. To do that, there must be an entry in the Kerberos configuration file (krb5.conf) to map each remote host name to its correct realm name. This host name entered must exactly match the remote

location name (RMTLOCNAME). The remote location parameter displayed by the DSPRDBDIRE or DSPDDMF command must match the domain name in the krb5.conf file. The following figure shows an example of the DSPRDBDIRE display:

```
                     Display Relational Database Detail

Relational database . . . . . .  : RCHASXXX

Remote location:
 Remote location . . . . . . . . . : rchasxxx.rchland.ibm.com
  Type . . . . . . . . . . . . . . : *IP
 Port number or service name . . . : *DRDA
 Remote authentication method  . . :
  Preferred method . . . . . . . . : *KERBEROS
  Allow lower authentication . . . : *NOALWLOWER
Text   . . . . . . . . . . . . . . :

Relational database type   . . . . : *REMOTE




  Press Enter to continue.
   F3=Exit   F12=Cancel
```

Here is a portion of the corresponding krb5.conf file contents showing the domain name matching the remote location name (Note: The Display File (DSPF) command is used to display the configuration file contents):

```
DSPF STMF('/QIBM/UserData/OS400/NetworkAuthentication/krb5.conf')

[domain_realm]
; Convert host names to realm names. Individual host names may be
; specified. Domain suffixes may be specified with a leading period
; and will apply to all host names ending in that suffix.
 rchasxxx.rchland.ibm.com = REALM.RCHLAND.IBM.COM
```

Jobs using Kerberos must be restarted when configuration changes occur to the krb5.conf file.

Related concepts

Control language

Distributed database programming

Application server security in a TCP/IP network:

The TCP/IP server has a default security of user ID with clear-text password. This means that, as the server is installed, inbound TCP/IP connection requests must have at least a clear-text password accompanying the user ID under which the server job is to run.

The security can either be changed with the Change DDM TCP/IP Attributes (CHGDDMTCPA) command or under the **Network** → **Servers** → **TCP/IP** → **DDM server properties**

in iSeries Navigator. You must have *IOSYSCFG special authority to change this setting.

There are two settings that can be used for lower server security:

- PWDRQD (*NO)

  Password is not required.
- PWDRQD(*VLDONLY)

  Password is not required, but must be valid if sent.

The difference between *NO and *VLDONLY is that if a password is sent from a client system, it is ignored in the *NO option. In the *VLDONLY option, however, if a password is sent, the password is validated for the accompanying user ID, and access is denied if incorrect.

Encrypted password required or PWDRQD(*ENCRYPTED) and Kerberos or PWDRQD(*KERBEROS) can be used for higher security levels. If Kerberos is used, user profiles must be mapped to Kerberos principles using Enterprise Identity Mapping (EIM).

The following example shows the use of the Change DDM TCP/IP Attributes (CHGDDMTCPA) command to specify that an encrypted password must accompany the user ID. To set this option, enter:

```
CHGDDMTCPA PWDRQD(*ENCRYPTED)
```

Note: The DDM/DRDA TCP/IP server was enhanced in V4R4 to support a form of password encryption called password substitution. In V4R5, a more widely-used password encryption technique, referred to as the Diffie-Hellman public key algorithm was implemented. This is the DRDA standard algorithm and is used by the most recently released IBM DRDA application requestors. The older password substitute algorithm is used primarily for DDM file access from PC clients. In V5R1 a 'strong' password substitute algorithm was also supported. The client and server negotiate the security mechanism that will be used, and any of the three encryption methods will satisfy the requirement of PWDRQD(*ENCRYPTED), as does the use of Secure Sockets Layer (SSL) datastreams.

Related concepts

Enterprise Identity Mapping (EIM)

Connection security protocols for DDM or DRDA:

Several connection security protocols are supported by the current DB2 UDB for iSeries implementation of distributed data management (DDM) or Distributed Relational Database Architecture (DRDA) over TCP/IP.

- User ID only
- User ID with clear-text password
- User ID with encrypted password
- Kerberos

With encrypted datastream support, the traditional communications trace support has little value. The Trace TCP/IP Application (TRCTCPAPP) command records outbound datastreams at a point before encryption, and inbound datastreams after decryption.

Secure Sockets Layer for DDM and DRDA:

DB2 Universal Database™ for iSeries Distributed Relational Database Architecture (DRDA) clients do not support Secure Sockets Layer (SSL).

However, similar function is available with Internet Protocol Security Protocol (IPSec).

The DDM TCP/IP server supports the SSL data encryption protocol. You can use this protocol to interoperate with clients such as iSeries Toolbox for Java™ and iSeries Access Family OLE DB Provider that support SSL for record-level access, and with any DDM file I/O clients provided by independent software vendors that might support SSL.

To use SSL with the iSeries DDM TCP/IP server, you must configure the client to connect to the well-known SSL port 448 on the server.

If you specify PWDRQD(*ENCRYPTED) on the Change DDM TCP/IP Attributes (CHGDDMTCPA) command on the server, you can use any valid password along with SSL. This is possible because the server recognizes that the whole datastream, including the password, is encrypted.

Related concepts

"Internet Protocol Security Protocol for DDM/DRDA"
Internet Protocol Security Protocol (IPSec) is a security protocol in the network layer that provides cryptographic security services. These services support confidential delivery of data over the Internet or intranets.

Secure Sockets Layer (SSL)

Connecting to System i: iSeries Access for Windows

Required programs:

You will need to set up and install SSL support.

iSeries server requirements:

For an iSeries server to communicate over SSL, it must be running V4R4 or later of the operating system, and have the following applications installed:

- TCP/IP Connectivity Utilities for i5, 5722-TC1 (Base TCP/IP support)
- IBM HTTP Server for i5/OS, 5722-DG1 (for access to Digital Certificate Manager)
- Digital Certificate Manager, 5722-SS1 - Boss Option 34

Internet Protocol Security Protocol for DDM/DRDA:

Internet Protocol Security Protocol (IPSec) is a security protocol in the network layer that provides cryptographic security services. These services support confidential delivery of data over the Internet or intranets.

On iSeries, IPSec, a component of the Virtual Private Networking (VPN) support, allows all data between two IP addresses or port combinations to be encrypted, regardless of application (such as DRDA or DDM). You can configure the addresses and ports that are used for IPSec. IBM suggests using port 447 for IPSec for either DRDA access or DDM access.

Use of any valid password along with IPSec will not in general satisfy the requirement imposed by specifying PWDRQD(*ENCRYPTED) on the Change DDM TCP/IP Attributes (CHGDDMTCPA) command at the server, because the application (DRDA or DDM) will not be able to determine if IPSec is being used. Therefore, you should avoid using PWDRQD(*ENCRYPTED) with IPSec.

Related concepts

"Secure Sockets Layer for DDM and DRDA" on page 57
DB2 Universal Database for iSeries Distributed Relational Database Architecture (DRDA) clients do not support Secure Sockets Layer (SSL).

Virtual Private Networking (VPN)

Considerations for certain passwords being passed as clear text:

Although iSeries supports the encryption of connection passwords, one of the connection security options you can specify in setting up an RDB directory entry is *USRIDPWD.

See the Add RDB Directory Entry command and the Change Relational Database Directory Entry command in Working with the relational database directory for more information.

If the server to which the connection is made allows the *USRIDPWD security option, the connection password can flow unencrypted. In V5R3, the SQL SET ENCRYPTION PASSWORD statement and the

ENCRYPT function can also cause passwords to flow over the network unencrypted. Currently, there are two possible solutions for encrypting datastreams. One is to use IPSec. As the other possibility, if you are using an AR that supports SSL, you can use that protocol to encrypt data transmitted to and from an iSeries AS.

Ports and port restrictions for DDM/DRDA:

With the advent of new choices for security of distributed data management (DDM) communications, the iSeries server administrator can restrict certain communications modes by blocking the ports they use. This topic discusses some of these considerations.

The DDM or DRDA TCP/IP server listens on port 447 (the well-known DDM port) and 446 (the well-known DRDA port) as well as 448 (the well-known SSL port). The DB2 Universal Database for iSeries implementation of DDM does not distinguish between the two ports 446 and 447, however, so both DDM and DRDA access can be done on either port.

Using the convention recommended for IPSec, the port usage for the DDM TCP/IP server follows:

- 446 for clear text datastreams
- 447 for IPSec encrypted datastreams (suggested)
- 448 for SSL encrypted datastreams (required)

You can block usage of one or more ports at the server by using the Configure TCP/IP (CFGTCP) command. To do this, choose the 'Work with TCP/IP port restrictions' option of that command. You can add a restriction so that only a specific user profile other than the one that QRWTLSTN runs under (normally QUSER) can use a certain port, such as 446. That effectively blocks 446. If 447 were configured for use only with IPSec, then blocking 446 would allow only encrypted datastreams to be used for DDM and DRDA access over native TCP/IP. You could block both 447 and 448 to restrict usage only to SSL. It might be impractical to follow these examples for performance or other reasons (such as current limited availability of SSL-capable clients), but they are given to show the possible configurations.

DDM server access control exit program for additional security

Customers who use menu-level security, which is accomplished by restricting the user's access to functions on the server, are likely to have a large number of public files. *Public files* are those files to which the public has some or all authority. A user exit program allows you to restrict each DDM user's access to public files and to private files.

The name of the program must be specified on the DDMACC parameter of the Change Network Attributes (CHGNETA) command.

User exit programs also let you block or filter DDM connection requests. All connection requests made by a DDM source system can be denied, or access to selected users can be granted. The user exit program must exist on the target server. The target DDM support calls this program:

- For each user's *initial* reference to a file to verify whether the user can have access to the file. When a file is referred to for I/O operations, this verification occurs only once, when the file is opened. The user exit program indicates to the TDDM whether the access request is accepted or rejected.
- For each DDM connection request.
- For each of the other functions listed in the *Subapplication* field of the table in Table 5 on page 61.

When a user exit program is specified, the TDDM first checks for errors in the access request that is received from the source server. If no errors are detected, the TDDM builds the parameter list, calls the user exit program, and passes the parameter list to it.

Related concepts

"Elements of security in an APPC network" on page 49
When Distributed Relational Database Architecture (DRDA) is used, the data resources of each server in the DRDA environment should be protected.

User exit program requirement

The purpose of the exit program created by the user is to determine whether a user's access request is to be accepted or rejected. It does so using the values that are passed to it in the parameter list.

The program can be written to verify all the values in the parameter list, or to verify part of them. The program *must* return a return code of 1 to indicate that the request is accepted, and it *should* return a 0 to indicate that the request is rejected.

The user exit program processes on the target DDM or DRDA server and must be located in a library in the system database (SYSBAS) if the target server is using independent auxiliary storage pools (independent ASPs).

User exit program parameter list for DDM

The user exit program on the target server passes two parameter values: a character return code field and a character data structure containing various parameter values.

The user exit program on the target server uses the character data structure parameter values, that are passed by the TDDM, to evaluate whether to allow the request from the source server. The parameter list is created each time a file access request or command request is sent to the TDDM; when any one of the functions shown for the *Subapplication* field is requested, the parameter list is created. When file I/O operations are performed, this parameter list is created only for the file open request, not for any of the I/O operation requests that follow.

The program uses the parameter list to determine whether a source server user's file access or command request should be accepted or rejected. The list contains the following parameters and values:

- The name of the user profile or default user profile under which the source server user's request is run.
- The name of the application program on the source server being used. For DDM use, the name is *DDM. For DRDA use, the name is *DRDA.
- The name of the command or function (subapplication) being requested for use on the target server or one of its files.

  Most of the functions listed in the following table directly affect a file, including the EXTRACT function, which extracts information from the file when commands such as Display File Description (DSPFD) or Display File Field Description (DSPFFD) are specified by the source server user. Some functions are member-related functions, such as the CHGMBR function, which allows characteristics of a member to be changed. The COMMAND function indicates that a command string is submitted by the Submit Remote Command (SBMRMTCMD) command to run on the target server. The SQLCNN function specifies a DRDA connect attempt.
- The name of the file (object) to be accessed in the way specified on the previous parameter. This field does not apply if a command string (COMMAND) or stream and directory access commands are being submitted or if it is a DRDA command.
- If the stream and directory access commands are specified, then the object and directory fields have a value of *SPC. The user must go to the *Other* field to get the alternative object name and alternative path name.
- The name of the library containing the file, if a file is being accessed.
- The name of the file member, if a file member is being accessed. Stream and access commands have a value of *N.
- The format field does not apply for DDM or DRDA.
- Depending on how the next field is used, the length varies.

- The *Other* field is used for as many as three of the following six values; the first two are always specified (*N might be used for the second value if the system name cannot be determined), and either of the last four might be specified, depending on the type of function specified in the *Subapplication* field.
  - The location name of the source server. This matches the RMTLOCNAME parameter value specified in the target server's device description for the source server if APPC communications is being used.
  - The system name of the source server.
  - If a file was specified and it is to be opened, (OPEN) for I/O operations, this field indicates which type of operation is being requested. For example, if a file is being opened for read operations only, the input request value is set to a 1 and the remaining values are set to a 0.
  - The alternative object name.
  - The alternative directory name.
  - The name of the iSeries command, if a command string is being submitted, followed by all of its submitted parameters and values.

Table 5. Parameter list for user exit program on target server

Field	Type	Length	Description
User	**Character**	**10**	**User profile name of target DDM job.**
Application	**Character**	**10**	**Application name:** • '*DDM ' for Distributed Data Management.
Subapplication	**Character**	**10**	**Requested function:** • 'ADDMBR ' 'DELETE ' 'RGZMBR ' • 'CHANGE ' 'EXTRACT ' 'RMVMBR ' • 'Change Data Area (CHGDTAARA) ' 'INITIALIZE' 'RNMMBR ' • 'CHGMBR ' 'LOAD ' 'Retrieve Data Area (RTVDTAARA)' • 'CLEAR ' 'LOCK ' 'SNDDTAQ ' • 'CLRDTAQ ' 'Move (MOVE) ' • 'COMMAND ' 'OPEN ' • 'Copy (COPY) ' 'RCVDTAQ ' • 'CREATE ' 'RENAME ' • 'SQLCNN '
Object	**Character**	**10**	**Specified file name. *N is used when the subapplication field is 'COMMAND '. *SPC is used when the file is a document or folder.**
	Character	**10**	**Specified library name. *N is used when the subapplication field is 'COMMAND '. *SPC is used when the library is a folder.**
Member	**Character**	**10**	**Specified member name. *N is used when the member name is not applicable.**
Format	**Character**	**10**	**Not applicable for DDM.**
Length	**Decimal**	**5,0**	**Length of the next field.**
Source Remote Location	**Character**	**10**	**Remote location unit name of source system (if SNA).**
Source System Name	**Character**	**10**	**System name of remote server. If this value is not available, this field contains '*N '.**

Table 5. Parameter list for user exit program on target server (continued)

Field	Type	Length	Description
Other	**Character**	**2000**	The use of this 2000 byte area depends on the request function. If it is SQLCNN, then the DRDA mapping should be used. For other functions, use the DDM mapping. **To use DDM:** The following varies, depending on the function. If OPEN is specified to open a file: **1** Input request Char(1) 1=yes 0=no **1** Output request Char(1) 1=yes 0=no **1** Update request Char(1) 1=yes 0=no **1** Delete request Char(1) 1=yes 0=no **12** Alternative object name. **63** Alternative directory name. **1921** The command string if COMMAND is specified to submit a command. **To use DRDA:** **9** Type definition name of DRDA application requester. Product ID of DRDA application requester. **3** Product code. **2** Version ID. **2** Release ID. **1** Modification level. **1983** Reserved

Note:

***N =** Null value indicates a parameter position for which no value is being specified, allowing other parameters to follow it in positional form.

User exit program example for DDM

This user exit program represents the source code for a program that is created by a security officer on a remote system in Chicago.

To define this user exit program to the server, the security officer specifies the following statement:

```
CHGNETA DDMACC(DJWLIB/$UEPGM)
```

where DJWLIB/$UEPGM is the qualified name of the user exit program.

Because the security officer wants to specifically prevent user KAREN from opening file RMTFILEX, the user exit program returns a 0 in the return code field when she attempts to open file RMTFILEX; the user exit program returns a 1 in the return code field in all other cases indicating that requests by other users are permitted.

Note: By using the code examples, you agree to the terms of the "Code license and disclaimer information" on page 209.

```
$UEPGM: PROCEDURE (RTNCODE,CHARFLD);
DECLARE
       RTNCODE  CHAR(1);
DECLARE
 1  CHARFLD,
    2  USER     CHAR(10),
    2  APP      CHAR(10),
    2  FUNC     CHAR(10),
    2  OBJECT   CHAR(10),
    2  DIRECT   CHAR(10),
    2  MEMBER   CHAR(10),
    2  RESERVED CHAR(10),
    2  LNGTH    PIC '99999',
    2  LUNAME   CHAR(10),
    2  SRVNAME  CHAR(10),
    2  OTHER,
       3  INRQS    CHAR(1),
       3  OUTRQS   CHAR(1),
       3  UPDRQS   CHAR(1),
       3  DELRQS   CHAR(1),
       3  ALTOBJ   CHAR(12),
       3  ALTDIR   CHAR(63),
       3  REMAING  CHAR(1921);
DECLARE
  OPEN   CHAR(10) STATIC INIT('OPEN'),
  KAREN  CHAR(10) STATIC INIT('KAREN'),
  RMTFILEX CHAR(10) STATIC INIT('RMTFILEX');
DECLARE
  ZERO  CHAR(1)  STATIC INIT('0'),
  ONE   CHAR(1)  STATIC INIT('1');
IF (FUNC   = OPEN    ) &
   (USER   = KAREN   ) &
   (OBJECT = RMTFILEX)
THEN
  RTNCODE = ZERO;
ELSE
  RTNCODE = ONE;
END $UEPGM;
```

Parameter list example for DDM

The commands in this topic are in a CL program that a user named KAREN on the source server (NEWYORK) is using. The remote location configuration of the target server (CHICAGO) specifies SECURELOC(*YES) for the NEWYORK source server. This action indicates that user IDs are to be sent and that a user profile for KAREN exists on the target server.

The program used by KAREN accesses a DDM file named LOCFILEX that opens a remote file named RMTFILEX on the target server in Chicago. Both servers are iSeries servers. The file is being opened for input.

```
CRTDDMF  FILE(LOCFILEX)  RMTFILE(LIBX/RMTFILEX)
         RMTLOCNAME(CHICAGO)

Open Database File (OPNDBF)  FILE(LOCFILEX)  OPTION(*INP)
Monitor Message (MONMSG)  MSGID(CPF0000)    EXEC(GOTO EXIT)

CLOF  OPNID(LOCFILEX)
EXIT:  End Program (ENDPGM)
```

When the Open Database File (OPNDBF) command is run on the NEWYORK source server, the DDM file named LOCFILEX is opened. DDM sends a request to the target server to open RMTFILEX in LIBX for input operations. From this information, the target server builds the following parameter list to be used by the user exit program for verification:

```
KAREN *DDM OPEN RMTFILEX LIBX *N 0 24 CHICAGO NEWYORK 1000
```

This parameter list shows only the significant characters that would be sent in each field; all the padded blanks and zeros are not shown. For example, the field containing KAREN is padded with five blanks because it is a 10-character field. This parameter list is sent only for the open operation, although several input operations might be performed on RMTFILEX.

This parameter list is sent to the user exit program specified on the DDMACC parameter of the Change Network Attributes (CHGNETA) command. The user exit program determines if user KAREN is authorized to open RMTFILEX. If she is authorized, the program returns a 1 in the return code field, and she can open the file and perform read operations. If the program returns a 0 in the return code field, user KAREN receives a message in the job log indicating that she is not authorized to use the file.

When all the input operations are completed, the Close File (CLOF) command runs on the source server, and DDM sends the request to close the file.

DRDA server access control exit programs with example

A security feature of the DRDA server, for both APPC and TCP/IP use, extends the use of the DDMACC parameter of the CHGNETA command to DRDA.

The parameter previously applied only to DDM file I/O access. The DRDA usage of the function is limited to connection requests, however, and not to requests for data after the connection is made.

If you do not choose to take advantage of this security function, you normally do not need to do anything. The only exception is if you are currently using a DDM exit program that is coded to reject operations if an unknown function code is received, and you are also using DRDA to access data on that server. In this case, you must modify your exit program so that a '1' is returned to allow DRDA access if the function code is 'SQLCNN '.

To use the exit program for blocking or filtering DRDA connections, you need to create a new DDM exit program, or modify an existing one.

This security enhancement includes a DRDA function code on the list of request functions that can be input to the program in the input parameter structure. The function code, named 'SQLCNN ' (SQL connect request), indicates that a DRDA connection request is being processed (see the FUNC parameter in the following example). The APP (application) input parameter is set to `'*DRDA      '` instead of `'*DDM       '` for DRDA connection request calls.

In addition to this enhancement, the following parameters are useful for DRDA:

- The USER parameter, allows the program to allow or deny DRDA access based on the user profile ID.
- The SRVNAME parameter in the following example might also be of use. If this parameter is set, it indicates the name of the client server. If it is not set, it has the value *N. It should always be set for an iSeries DRDA Application Requester.
- The TYPDEFN gives additional information about the type of client attempting to connect.
- The PRDID (product ID) parameter identifies the product that is attempting to connect, along with the product's release level. A partial list of these codes follows. (You should verify the non-IBM codes before you use them in an exit program.)

  QSQ IBM DB2 Universal Database for iSeries

  DSN IBM DB2 Universal Database for z/OS®

  SQL IBM DB2 Universal Database for Linux®, UNIX® and Windows® (formerly called DDCS)

  ARI IBM DB2 Universal Database for VSE and VM

  GTW Oracle Corporation products

  GVW Grandview DB/DC Systems products

  XDB XDB Systems products

IFX Informix® Software products

RUM Wall Data Rumba for Database Access

SIG StarQuest products

STH FileTek products

The rest of the field is structured as `vvrrm`, where `vv` is version, `rr` is release, and `m` is modification level.

The *DDM Architecture Reference* manual and the *DRDA Reference* (both available from The Open Group) give more information about these fields.

If the exit program returns a RTNCODE value of '0', and the Application Requester system type is iSeries, then the message indicating the connection failure to the user will be SQ30060, 'User is not authorized to relational database ....'. In general, the response to a denial of access by the exit program is the DDM RDBATHRM reply message, which indicates that the user is not authorized to the relational database.

Restrictions

If a function check occurs in the user exit program, the same reply message will be returned, and the connection attempt will fail. The exit program must not do any committable updates to DB2 UDB for iSeries, or unpredictable results might occur. A further restriction results from the fact that when the prestart jobs used with the TCP/IP server are recycled for subsequent use, some cleanup is done to the jobs for security reasons. Part of this processing involves the use of the RCLACTGRP ACTGRP(*ELIGIBLE) function. As a result, attempts to use any residual linkages in the prestart server job to activation groups destroyed by the RCLACTGRP can result in MCH3402 exceptions (where the program tried to refer to all or part of an object that no longer exists). Furthermore, an exit program should not attempt to access a file that was opened in a prior invocation of the prestart server job.

Example

This example demonstrates a PL/I user exit program that allows all DDM operations, and all DRDA connections except for when the user ID is 'ALIEN'.

Note: By using the code examples, you agree to the terms of the "Code license and disclaimer information" on page 209.

```
/*****************************************************************/
/*                                                               */
/* PROGRAM NAME: UEPALIEN                                        */
/*                                                               */
/* FUNCTION:     USER EXIT PROGRAM THAT IS DESIGNED TO           */
/*               RETURN AN UNSUCCESSFUL RETURN CODE WHEN         */
/*               USERID 'ALIEN' ATTEMPTS A DRDA CONNECTION.      */
/*               IT ALLOWS ALL TYPES OF DDM OPERATIONS.          */
/*                                                               */
/* EXECUTION:    CALLED WHEN ESTABLISHED AS THE USER EXIT        */
/*               PROGRAM.                                        */
/*                                                               */
/* ALL PARAMETER VARIABLES ARE PASSED IN EXCEPT:                 */
/*                                                               */
/*    RTNCODE - USER EXIT RETURN CODE ON WHETHER FUNCTION IS     */
/*              ALLOWED: '1' INDICATES SUCCESS; '0' FAILURE.     */
/*                                                               */
/*****************************************************************/

UEPALIEN: PROCEDURE (RTNCODE,CHARFLD);

DECLARE RTNCODE CHAR(1);          /* DECLARATION OF THE EXIT     */
```

```
                                /* PROGRAM RETURN CODE.  IT     */
                                /* INFORMS REQUEST HANDLER      */
                                /* WHETHER REQUEST IS ALLOWED.  */
DECLARE                         /* DECLARATION OF THE CHAR      */
 1  CHARFLD,                    /* FIELD PASSED IN ON THE CALL. */
    2  USER     CHAR(10),       /* USER PROFILE OF DDM/DRDA USER*/
    2  APP      CHAR(10),       /* APPLICATION NAME             */
    2  FUNC     CHAR(10),       /* REQUESTED FUNCTION           */
    2  OBJECT   CHAR(10),       /* FILE NAME                    */
    2  DIRECT   CHAR(10),       /* LIBRARY NAME                 */
    2  MEMBER   CHAR(10),       /* MEMBER NAME                  */
    2  RESERVED CHAR(10),       /* RESERVED FIELD               */
    2  LNGTH    PIC '99999',    /* LENGTH OF USED SPACE IN REST */
    2  REST,                   /* REST OF SPACE = CHAR(2000)   */
      3  LUNAME  CHAR(10),      /* REMOTE LU NAME (IF SNA)      */
      3  SRVNAME CHAR(10),      /* REMOTE SERVER NAME           */
      3  TYPDEFN CHAR(9),       /* TYPE DEF NAME OF DRDA AR     */
      3  PRDID,                 /* PRODUCT ID OF DRDA AR        */
        5 PRODUCT  CHAR(3),     /* PRODUCT CODE                 */
        5 VERSION  CHAR(2),     /* VERSION ID                   */
        5 RELEASE  CHAR(2),     /* RELEASE ID                   */
        5 MOD      CHAR(1),     /* MODIFICATION LEVEL           */
      3  REMAING CHAR(1983);    /* REMAINING VARIABLE SPACE.    */

START:
IF (USER = 'ALIEN' &            /* IF USER IS 'ALIEN' AND       */
      FUNC = 'SQLCNN') THEN     /*   FUNCTION IS DRDA CONNECT   */
  RTNCODE = '0';                /* SET RETURN CODE TO UNSUCCESSFUL*/
ELSE                            /* IF ANY OTHER USER, OR DDM    */
  RTNCODE = '1';                /* SET RETURN CODE TO SUCCESSFUL  */

END UEPALIEN;
```

User exit program considerations for DDM

There are some considerations that you should understand before using user exit programs for DDM.

If the user exit program is a CL program that creates an i5/OS exception, an inquiry message is sent to the server operator on the target server if, for the target job, the job attribute INQMSGRPY is *RQD (the default) or *SYSRPYL with no value in the reply list for this message. The user exit program waits for a response to the message on the target server, which causes the source job to wait also.

There are other potential situations in which waiting could occur. For example, if lengthy wait values are specified on the WAIT parameter of the Allocate Object (ALCOBJ) or Receive Message (RCVMSG) command, both the source and target jobs wait up to the maximum time specified for an object lock to be obtained or a message to be received by the target job.

Related concepts

"Performance considerations for DDM" on page 134
These topics provide information to help you improve performance when using DDM and also provide some information about when to use something other than DDM to accomplish some functions.

Use CL and DDS with DDM

This topic contains DDM-related information about specific iSeries control language (CL) commands, data description specifications (DDS) considerations, DDS keywords, and DDM user profile authority.

Related concepts

"iSeries server as the source server for DDM" on page 13
When an application program or user in a source server job first refers to a DDM file, several actions occur as part of processing the request on the source server.

Control language

Related reference

"CL command considerations for DDM" on page 32
Both compiled control language (CL) programs and interactively entered CL commands can refer to DDM files.

"DDM file requirements" on page 44
Before remote files can be accessed by an iSeries server, DDM files must be created on the source server.

"Reclaim DDM resources (RCLRSC and RCLDDMCNV commands)" on page 122
When an iSeries user wants to ensure that the resources for all APPC conversations (including DDM conversations) that are no longer active are returned to the server, the Reclaim Resources (RCLRSC) command can be used.

DDM-specific CL commands

This topic covers DDM-specific CL commands.

CHGDDMF (Change DDM File) command

The Change DDM File (CHGDDMF) command changes one or more of the attributes of a DDM file on the local (source) server.

The DDM file is used as a reference file by programs on the iSeries source server to access files located on any target server in the i5/OS DDM network.

To use this command, you can enter the command as shown in the following example or select option 2 (Change DDM File) from the Work with DDM Files display.

Related reference

"WRKDDMF (Work with DDM Files) command" on page 74
The Work with DDM Files (WRKDDMF) command allows you to work with existing DDM files from a list display. From the list display, you can change, delete, display, or create DDM files.

Change DDM File (CHGDDMF) command

Example: CHGDDMF command:

This command changes the communications mode for the DDM file named SALES stored in the SOURCE library on the source server; the mode is changed to MODEX.

```
CHGDDMF  FILE(SOURCE/SALES)  MODE(MODEX)
```

CRTDDMF (Create DDM File) command

The Create DDM File (CRTDDMF) command creates a DDM file on the local (source) server.

The DDM file is used as a reference file by programs on an iSeries server to access files located on any remote (target) server in the iSeries DDM network. Programs on the local iSeries server know a remote file only by the DDM file's name, not the remote file's actual name. (The DDM file name, however, can be the same as the remote file name.)

The DDM file is also used when a CL command is submitted to the remote server. (The Submit Remote Command (SBMRMTCMD) command is used to submit the CL command, and the remote server must be an iSeries server or a System/38.) When the SBMRMTCMD command is being used, the remote file normally associated with the DDM file is ignored.

The DDM file contains the name of the remote file being accessed and the remote location information that identifies a remote (target) server where the remote file is located. It can also specify other attributes that are used to access records in the remote file.

To use this command, you can enter the command as shown in the following examples or select F6 (Create DDM file) from the Work with DDM Files display.

Related concepts

APPC, APPN, and HPR

Related reference

"WRKDDMF (Work with DDM Files) command" on page 74
The Work with DDM Files (WRKDDMF) command allows you to work with existing DDM files from a list display. From the list display, you can change, delete, display, or create DDM files.

Create DDM File (CRTDDMF) command

Examples: CRTDDMF command:

These examples display different uses for the CRTDDMF command.

Example: Create a DDM file to access a file on a System/38

```
CRTDDMF  FILE(SOURCE/SALES)  RMTFILE(*NONSTD 'SALES.REMOTE')
   RMTLOCNAME(NEWYORK)
```

This command creates a DDM file named SALES and stores it in the SOURCE library on the source server. This DDM file uses the remote location NEWYORK to access a remote file named SALES stored in the REMOTE library on a System/38 in New York.

Example: Create a DDM file to access a file member on an iSeries server

```
CRTDDMF  FILE(SOURCE/SALES)  RMTLOCNAME(NEWYORK)
   RMTFILE(*NONSTD 'REMOTE/SALES(APRIL)')
```

This command creates a DDM file similar to the one in the previous example, except that now it accesses the member named APRIL in the remote SALES file stored in the REMOTE library on an iSeries server.

Example: Create a DDM file to access a file on a System/36

```
CRTDDMF  FILE(OTHER/SALES)  RMTFILE(*NONSTD 'PAYROLL')
   RMTLOCNAME(DENVER)  LVLCHK(*NO)
```

This command creates a DDM file named SALES, and stores it in the library OTHER on the source server. The remote location DENVER is used by the DDM file to access a remote file named PAYROLL on a System/36 in Denver. No level checking is performed between the PAYROLL file and the application programs that access it. Because the ACCMTH parameter was not specified, the access method for the target server is selected by the source iSeries server when the DDM file is opened to access the remote file.

DSPDDMF (Display DDM Files) command

The Display DDM Files (DSPDDMF) command displays the details of a DDM file.

To use this command, you can type the command or select option 5 (Display details) from the Work with DDM Files display.

Related reference

"WRKDDMF (Work with DDM Files) command" on page 74
The Work with DDM Files (WRKDDMF) command allows you to work with existing DDM files from a list display. From the list display, you can change, delete, display, or create DDM files.

Display DDM File (DSPDDMF) command

RCLDDMCNV (Reclaim DDM Conversations) command

The Reclaim DDM Conversations (RCLDDMCNV) command is used to reclaim all DDM source server conversations that are not currently being used by a source job.

The conversations are reclaimed even if the value of the job's DDMCNV attribute is *KEEP, or if the command is entered within an activation group. The command allows the user to reclaim unused DDM conversations without closing all open files or doing any of the other functions performed by the Reclaim Resources (RCLRSC) command.

The RCLDDMCNV command applies only to the DDM conversations for the job on the *source* server in which the command is entered. For each DDM conversation used by the source job, there is an associated job on the target server; the target job ends automatically when the associated DDM conversation ends.

Although this command applies to *all* DDM conversations used by a job, using it does *not* mean that all of them will be reclaimed. A conversation is reclaimed *only* if it is not being actively used.

Related reference

"Control DDM conversations" on page 120
Normally, the DDM conversations associated with a source server job are kept active until one of the conditions described in this topic is met.

Reclaim DDM Conversations (RCLDDMCNV) command

SBMRMTCMD (Submit Remote Command) command

The Submit Remote Command (SBMRMTCMD) command submits a command using DDM to run on the target server.

The remote location information in the DDM file is used to determine the communications line to be used, and thus, indirectly identifies the target server that is to receive the submitted command.

You can use the SBMRMTCMD command to send commands to any of the following target servers:

- iSeries
- System/38
- Any server that supports the Submit System Command (SBMSYSCMD) DDM command

The SBMRMTCMD command can be used to send CL commands (and only CL) to an iSeries server or a System/38. It can also be used to send commands to target servers other than iSeries or System/38 servers if the target server supports the DDM architecture Submit System command. The command must be in the syntax of the target server. The SBMRMTCMD command cannot be used to send operation control language (OCL) commands to a System/36 target because the System/36 server does not support the function.

The primary purpose of this command is to allow a user or program using the source server to perform file management operations and file authorization activities on files located on a target server. The user must have the proper authority for the target server objects that the command is to operate on. The following actions are examples of what can be performed on remote files using the SBMRMTCMD command:

- Create or delete device files
- Grant or revoke object authority to remote files
- Verify files or other objects
- Save or restore files or other objects

Although the command can be used to do many things with files or objects, some are not as useful as others. For example, you can use this command to display the file descriptions or field attributes of remote files, or to dump files or other objects, but the output remains at the target server. Another way to display remote file descriptions and field attributes at the source system is to use the Display File Description (DSPFD) and Display File Field Description (DSPFFD) commands. Specify the SYSTEM(*RMT) parameter and the names of the DDM files associated with the remote files. This returns the information you want directly to the local server.

A secondary purpose of this command is to allow a user to perform nonfile operations (such as creating a message queue) or to submit user-written commands to run on the target server. The CMD parameter allows you to specify a character string of up to 2000 characters that represents a command to be run on the target server.

Related concepts

"Parts of DDM: DDM file" on page 8
A system object with type *FILE exists on the source server to identify a remote file. It combines the characteristics of a device file and a database file. As a device file, the DDM file refers to a remote location name, local location name, device name, mode, and a remote network ID to identify a remote server as the target server. The DDM file appears to the application program as a database file and serves as the access device between a source server program and a remote file.

"Source server actions dependent on type of target server" on page 17
If the target server is not another iSeries server or System/38, only the DDM architecture commands defined in Level 2.0 and earlier of the DDM architecture are used.

Related reference

"Perform file management functions on remote servers" on page 119
i5/OS DDM supports creating, deleting, or renaming of files on a remote server.

Reclaim DDM Conversations (RCLDDMCNV) command

iSeries and System/38 target systems on the SBMRMTCMD command:

The SBMRMTCMD command can submit any CL command that can run in both the batch environment and using the QCAEXEC server program.

That is, a command can be submitted using the SBMRMTCMD command if it has both of the following values for the ALLOW attribute:

***BPGM**
The command can be processed in a compiled CL program that is called from batch entry.

***EXEC**
The command can be used as a parameter on the CALL command and get passed as a character string to the server program for processing.

You can look for these possible values using the Display Command (DSPCMD) command. (The SBMRMTCMD command uses the QCAEXEC or QCMDEXEC system program to run the submitted commands on the target server.) However, because some of these allowable commands require intervention on the target server and might not produce the results expected, you should consider the items listed in the topic Restrictions for the SBMRMTCMD first.

The user must have the proper authority for both the CL command being submitted and for the target server objects that the command is to operate on.

Related reference

"Restrictions for the SBMRMTCMD command"
This topic describes restrictions for the SBMRMTCMD command.

"DDM-related CL command summary charts" on page 104
This topic shows summary charts containing most of the control language (CL) commands used with DDM.

Restrictions for the SBMRMTCMD command:

This topic describes restrictions for the SBMRMTCMD command.

- Although remote file processing is synchronous within the user's job, which includes two separate jobs (one running on each server), file processing on the target server operates independently of the source server. Commands such as Override with Database File (OVRDBF), Override with Message File

(OVRMSGF), and Delete Override (DLTOVR) that are dependent on the specific position of a program in a program stack (*recursion level*) or request level might *not* function as expected.

For example, when multiple recursion levels that involve overrides at each level occur on the source server, and one or more overrides at a given level are submitted to the target server on the SBMRMTCMD command, the target server job has no way of knowing the level of the source server job. That is, a target server override can still be in effect after the source server override for a particular recursion level has ended.

- Output (such as spooled files) created by a submitted command exists only on the target server. The output is *not* sent back to the source server.
- Some types of CL commands should *not* be submitted to a target iSeries server. The following items are examples of types that are *not* the intended purpose of the SBMRMTCMD command and that might produce undesirable results:
  - All of the OVR*xxx*F commands that refer to database files, message files, and device files (including communications and save files).
  - All of the DSP*xxxx* commands, because the output results remain at the target server.
  - Job-related commands like Reroute Job (RRTJOB) that are used to control a target server's job. The Change Job (CHGJOB) command, however, *can* be used.
  - Commands that are used to service programs, like Service Job (SRVJOB), Trace Job (TRCJOB), Trace Internal (TRCINT), or Dump Job (DMPJOB).
  - Commands that might cause inquiry messages to be sent to the system operator, like Start Printer Writer (STRPRTWTR). (Pass-through can be used instead.)
  - Commands that attempt to change the Independent Auxiliary Storage Pool (ASP) of the target job (for example, SETASPGRP) should not be issued using Submit Remote Command.
- Translation is not performed for any *immediate* messages created by the target server, because they are not stored on the server; the text for an immediate message is sent directly to the source server to be displayed. (For all other message types, the target server sends back a message identifier; the message text that exists on the source server for that message identifier is the text that is displayed. This message text is whatever the source server text has been translated to.)
- A maximum of 10 messages, created during the running of a submitted command, can be sent by the target server to the source server. If more than 10 messages are created, an additional *informational* message is sent that indicates where the messages exist (such as in a job log) on the target server. If one of those messages is an *escape* message, the first nine messages of other types are sent, followed by the informational message and the escape message.
- The only types of messages that are sent by the target server are completion, informational, diagnostic, and escape messages.

Related reference

"iSeries and System/38 target systems on the SBMRMTCMD command" on page 70
The SBMRMTCMD command can submit any CL command that can run in both the batch environment and using the QCAEXEC server program.

Reclaim DDM Conversations (RCLDDMCNV) command

Examples: SBMRMTCMD command:

The examples display different uses for the SBMRMTCMD command.

Example: Submit a command to create another DDM file on the remote server

```
SBMRMTCMD  CMD('CRTDDMF  FILE(SALES/MONTHLY)
   RMTFILE(*NONSTD ''SALES/CAR(JULY)'')
   RMTLOCNAME(DALLAS)')  DDMFILE(CHICAGO)
```

This submitted command creates, on the target server identified by the information in the DDM file named CHICAGO, another DDM file named MONTHLY; the new DDM file is stored in a library named

SALES on the server defined by DDMFILE CHICAGO. The new DDM file on the CHICAGO server is used to access a file and *member* on a different server named DALLAS. The accessed file is named CAR in the library SALES and the member name in the file is JULY.

Notice that this CRTDDMF command string contains *three* sets of single quotation marks: one set to enclose the entire command being submitted (required by the CMD parameter on the SBMRMTCMD command), and a double set to enclose the file and member named in the RMTFILE parameter. Because the use of *NONSTD requires that nonstandard file names be enclosed in a set of single quotation marks, this second set of single quotation marks must be doubled because it is within the first set of single quotation marks.

Example: Submit a command to change text in a display file

```
SBMRMTCMD  CMD('CHGDSPF  FILE(LIBX/STANLEY)
   TEXT(''Don''''t forget to pair apostrophes.'')')
   DDMFILE(SMITH)
```

This command changes the text in the description of the display device file named STANLEY stored in library LIBX. Because the submitted command requires an outside set of single quotation marks (for the CMD parameter), each single quotation mark (') or quotation marks ('') normally required in the TEXT parameter for *local* server processing must be doubled again for *remote* server processing. The preceding coding produces a single quotation mark in the text when it is displayed or printed on the remote server.

Example: Submit a command to replace a library list on the remote server

```
SBMRMTCMD  CMD('CHGLIBL  LIBL(QGPL QTEMP SALES EVANS)')
   DDMFILE(EVANS)
```

This command changes the user's portion of the library list being used by the target job associated with the DDM file named EVANS, which is being used by the source job in which this SBMRMTCMD command is being submitted. In that source job, if there are other open DDM files that specify the remote location information, this library list is used for them also.

Additional considerations: SBMRMTCMD command:

This topic describes additional considerations for the SBMRMTCMD command.

Override use example

The DDMFILE parameter on the SBMRMTCMD command is used to determine which target server the command (CMD parameter) should be sent to. Overrides that apply to the DDM file (not the remote file) are taken into account for this function. For example, if a file override was in effect for a DDM file because of the following commands, which override FILEA with FILEX, then the target server that the Delete File (DLTF) command is sent to is the one associated with the remote location information specified in DDM FILEX (the values point to the DENVER system, in this case).

```
CRTDDMF  FILE(SRCLIB/FILEA)  RMTFILE(SALES/CAR)
   RMTLOCNAME(CHICAGO)
CRTDDMF  FILE(SRCLIB/FILEX)  RMTFILE(SALES/CAR)
   RMTLOCNAME(DENVER)
OVRDBF   FILE(FILEA)  TOFILE(SRCLIB/FILEX)
SBMRMTCMD  CMD('DLTF RMTLIB/FRED')  DDMFILE(SRCLIB/FILEA)
```

This SBMRMTCMD command deletes the file named FRED from the DENVER server.

DDM conversations

When a SBMRMTCMD command is run on the target server, it has a target server job associated with it. Successive SBMRMTCMD commands submitted using the same DDM file and DDM conversation might run in the same or different target server jobs, depending on the value of the DDMCNV job attribute. The

value of the DDMCNV job attribute determines whether the DDM conversation is dropped or remains active when the submitted function has completed. If the conversation is dropped, the next SBMRMTCMD command runs using a different target job. If several commands are submitted, either DDMCNV(*KEEP) should be in effect, or display station pass-through should be used instead of DDM.

Command syntax verifying

The syntax of the command character string being submitted by the CMD parameter is not verified by the source server. In the case of a user-defined command, for example, the command definition object might or might not exist on the source server.

Command running results

Because the submitted command runs as part of the target server's job, the attributes of that job (such as the library search list, user profile, wait times, and running priority) might cause a different result than if the command were run locally. If you find that you are having difficulty submitting a command and, for example, the reason is the target server uses a different library list, you can use the SBMRMTCMD command to edit the library list.

Error message handling

For errors detected by the target server when processing the submitted command, the source server attempts to send the same error information that was created on the target server to the user. However, if the source server does not have an equivalent message for the one created on the target server, the message sent to the source server user has the message identifier and is of the message type and severity that was created on the target server; the message text sent for the error is default message text.

If the target server is a system other than an iSeries server or System/36, messages sent to the source server have no message identifiers or message types. The only information received from such a target server is the message text and a severity code. When a high severity code is returned from the target server, the source server user receives a message that the SBMRMTCMD command ended abnormally. Other messages sent by the target server are received as informational with no message identifiers.

For example, you might see the following statements in your job log when both the source and target are iSeries servers:

```
INFO CPI9155 'Following messages created on target server.'
DIAG CPD0028 'Library ZZZZ not found.'
ESCP CPF0006 'Errors occurred in command.'
```

When a target server other than an iSeries server returns the same message to an iSeries source server, the job log looks like this:

```
INFO CPI9155 'Following messages created on target server.'
INFO nomsgid 'Library ZZZZ not found.'
INFO nomsgid 'Errors occurred in command.'
ESCP CPF9172 'SBMRMTCMD command ended abnormally.'
```

The target server messages can be viewed on the source server by using pass-through and either the Work with Job (WRKJOB) or Work with Job Log (WRKJOBLOG) command. If the target job ends, the messages are in the target server's output queue, where they can be displayed by the Work with Output Queue (WRKOUTQ) command.

If the SBMRMTCMD command is used to call a CL program on the target server, any escape message that is not monitored and is created by the program is changed into an inquiry message and is sent to the system operator. If you don't want the target system operator to have to respond to this inquiry message before the job can continue, you can refer to the CL topic in the iSeries Information Center and do either of the following items on the target server:

- If you want to specify a default reply for a specific *job*, you can use the INQMSGRPY parameter on either the Create Job Description (CRTJOBD) or Change Job Description (CHGJOBD) command to specify either *DFT or *SYSRPYL in the job description for the target job. You can also do the same thing if you use the SBMRMTCMD command to submit the Change Job (CHGJOB) command to the target server.
- If you want to specify a default reply message for a specific *inquiry message* in the job, you can use the Add Reply List Entry (ADDRPYLE) command (on the target server) to add an entry for that message to the system-wide automatic message reply list (SYSRPYL). Then, if INQMSGRPY(*SYSRPYL) is specified in the job description, this default reply can be sent whenever that inquiry message occurs in the job.

Independent auxiliary storage pools (ASPs)

If the target system has online independent ASPs, the independent ASP group of the target job is established when the conversation is started and might not be changed. User-defined or CL commands that attempt to change the independent ASP group of the target job (for example, SETASPGRP or DLTUSRPRF) might fail if submitted to a target system that has online independent ASPs.

Related concepts

"DDM-related jobs and DDM conversations" on page 20
This topic provides additional information about activation groups, source server jobs, target server jobs, and the DDM conversations used by those jobs.

Related reference

"DDMCNV parameter considerations" on page 97
The DDMCNV parameter is a job-related parameter that controls whether Advanced Program-to-Program Communication (APPC) or iSeries conversations in the job that is allocated for DDM use (that is, DDM conversations) are to be automatically dropped or kept active for the source job.

"OVRDBF (Override with Database File) command" on page 94
The Override with Database File (OVRDBF) command can be used with DDM to override (replace) a local database file named in the program with a DDM file; the DDM file causes the associated remote file to be used by the program instead of the local database file.

WRKDDMF (Work with DDM Files) command

The Work with DDM Files (WRKDDMF) command allows you to work with existing DDM files from a list display. From the list display, you can change, delete, display, or create DDM files.

For the following displays, it is assumed that you have created DDM files using theCreate DDM File (CRTDDMF) command. If you enter the WRKDDMF command and specify library `WILSON` and file `A`, the following display is shown:

```
                          Work with DDM Files

Position to  . . . . . .  _____________________

Type options, press Enter.
  1=Create DDM file   2=Change DDM file   4=Delete   5=Display details
  6=Print details

                                                                  Remote
Option   Local File              Remote File                      Location
  _       _____________________
  _       WILSON/A                 A                               S36
                                                                    Bottom
F3=Exit   F5=Refresh   F9=Print list   F12=Cancel
```

To *create* a DDM file using this display, type a 1 in the option column and type the names of the library and file you want to create, then press the Enter key. For example, type a 1 (Create DDM file) in the option field and WILSON/TEST in the local file column of the top list entry (as shown in the following display), and then press the Enter key. The Create DDM File display is shown.

```
                          Work with DDM Files

Position to  . . . . . .  _____________________

Type options, press Enter.
  1=Create DDM file   2=Change DDM file   4=Delete   5=Display details
  6=Print details

                                                                  Remote
Option   Local File              Remote File                      Location
  1       WILSON/TEST__________
  _       WILSON/A                 A                               S36
                                                                    Bottom
F3=Exit   F5=Refresh   F9=Print list   F12=Cancel
```

```
                        Create DDM File (CRTDDMF)

Type choices, press Enter.

DDM file . . . . . .     TEST        Name
  Library  . . . . . .     WILSON      Name, *CURLIB
Remote file:
  File . . . . . .                     Name, *NONSTD
    Library  . . . . . .               Name, *LIBL, *CURLIB
  Nonstandard file 'name'  . . .



Remote location:
  Name or address  . . . . . .



  Type . . . . . .   *SNA          *SNA, *IP

                                                                    More...
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display        F24=More keys
```

On the Create DDM File display, type the required values, and change or use the default values given. By pressing F10 (Additional parameters), you can page through the command parameters as they are shown on two displays. By pressing the Page Down key, you are shown these additional parameters:

```
                        Create DDM File (CRTDDMF)

Type choices, press Enter.

Text 'description' . . . . . .   *BLANK


                        Additional Parameters

Device:
  APPC device description  . . .   *LOC          Name, *LOC
Local location . . . . . .         *LOC          Name, *LOC, *NETATR
Mode . . . . . .                   *NETATR       Name, *NETATR
Remote network identifier  . . .   *LOC          Name, *LOC, *NETATR, *NONE
Port number  . . . . . .           *DRDA         *DRDA, 1-65535
Access method:
  Remote file attribute  . . . .   *RMTFILE      *RMTFILE, *COMBINED...
  Local access method  . . . . .                 *BOTH, *RANDOM, *SEQUENTIAL
Share open data path . . . . . .   *NO           *NO, *YES
Protected conversation . . . . .   *NO           *NO, *YES
                                                                    More...
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

```
                        Create DDM File (CRTDDMF)

Type choices, press Enter.

Record format level check  . . .   *RMTFILE      *RMTFILE, *NO
Authority  . . . . . . . . . . .   *LIBCRTAUT    Name, *LIBCRTAUT, *ALL...
Replace file . . . . . . . . . .   *YES          *YES, *NO

                                                                      Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

After you have typed in the values, press the Enter key to process the command and return to the Work with DDM Files display.

If you want to *change* a DDM file, type a 2 (Change DDM file) on the Work with DDM Files display next to the file that you want to change, or type the option number in the top list entry of the Options column and specify the local file that you want changed. For example, type a 2 (Change DDM file) in the *Option* column of the local file named WILSON/TEST.

```
                           Work with DDM Files

Position to  . . . . . .  ____________________

Type options, press Enter.
  1=Create DDM file   2=Change DDM file   4=Delete   5=Display details
  6=Print details

                                                                  Remote
Option   Local File              Remote File                      Location
  _       _____________________
  _       WILSON/A                 A                                S36
  2       WILSON/TEST              TESTFILE.TESTLIB                 S38

                                                                    Bottom
F3=Exit   F5=Refresh   F9=Print list   F12=Cancel
```

Press the Enter key and the Change DDM File display is shown.

For example, if you *only* want to add a text description, type in the description and press the Enter key. But, if you want to make additional changes, press F10 (Additional parameters), and you can page through the command parameters as they are shown on two displays.

```
                        Change DDM File (CHGDDMF)

Type choices, press Enter.

DDM file . . . . . .    TEST         Name
  Library  . . . . . .    WILSON       Name, *LIBL, *CURLIB
Remote file:
  File . . . . . .    *SAME         Name, *SAME, *NONSTD
    Library  . . . . . .                Name, *LIBL, *CURLIB
  Nonstandard file 'name'  . . .



Remote location:
  Name or address  . . . . . .   *SAME



  Type . . . . . .   *SAME         *SAME, *SNA, *IP
Record format level check  . . .   *SAME         *SAME, *RMTFILE, *NO
                                                                  More...
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

If you want to change the mode parameter, type in that value, and then press the Enter key.

```
                        Change DDM File (CHGDDMF)

 Type choices, press Enter.

 Text 'description' . . . . . .   *SAME


                           Additional Parameters

 Device:
   APPC device description  . . .   *SAME         Name, *SAME, *LOC
 Local location . . . . . .   *SAME         Name, *SAME, *LOC, *NETATR
 Mode . . . . . .   *SAME         Name, *SAME, *NETATR
 Remote network identifier  . . .   *SAME         Name, *SAME, *LOC, *NETATR...
 Port number  . . . . . .   *SAME         *SAME, *DRDA, 1-65535
 Access method:
   Remote file attribute  . . . .   *SAME         *SAME, *RMTFILE, *COMBINED...
   Local access method  . . . . .                *BOTH, *RANDOM, *SEQUENTIAL
 Share open data path . . . . . .   *SAME         *SAME, *NO, *YES
 Protected conversation . . . . .   *SAME         *SAME, *NO, *YES
                                                                  Bottom
 F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
 F24=More keys
```

After you press the Enter key, you return to the Work with DDM Files display.

If you want to *display* the details of a DDM file, type a 5 (Display details) on the Work with DDM Files display next to the file that you want to display, or type the option number in the top list entry of the Options column and specify the local file you want to display. For example, type a 5 (Display details) in the *Option* column and type `WILSON/TEST` in the *Local File* column of the top list entry.

You can also display the details of a file by using the Display DDM Files (DSPDDMF) command.

```
                          Work with DDM Files

  Position to  . . . . . _____________________

  Type options, press Enter.
    1=Create DDM file   2=Change DDM file   4=Delete   5=Display details
    6=Print details

                                                                      Remote
  Option   Local File              Remote File                        Location
    5       WILSON/TEST__________
    _       WILSON/A                 A                                 S36
    _       WILSON/TEST              TESTFILE.TESTLIB                  S38









                                                                       Bottom
  F3=Exit   F5=Refresh   F9=Print list   F12=Cancel
```

Press the Enter key and the Display Details of DDM File display is shown.

```
                     Display Details of DDM File         SYSTEM:  AS400B

Local file:
  File . . . . . . . . . . . . . . . . . . . . . :    TEST
    Library  . . . . . . . . . . . . . . . . . . :      WILSON

Remote file  . . . . . . . . . . . . . . . . . . :    TESTFILE.TESTLIB


Remote location:
  Remote location  . . . . . . . . . . . . . . . :    S38
  Device description . . . . . . . . . . . . . . :    *LOC
  Local location . . . . . . . . . . . . . . . . :    *LOC
  Remote location network ID . . . . . . . . . . :    *LOC
  Mode . . . . . . . . . . . . . . . . . . . . . :    S38MODE1




  Press Enter to continue.
                                                                      More...
  F3=Exit    F12=Cancel
```

Page down to see the second display.

```
                    Display Details of DDM File
                                                        SYSTEM:  AS400B

Access method
  Remote file attribute  . . . . :    *RMTFILE
  Local access method  . . . . . :

Share open data path . . . . . :    *NO
Check record format level ID . . :    *RMTFILE
Text . . . . . . . . . . . . . . :    TEST VERSION FOR DDM









  Press Enter to continue.
                                                                      Bottom
  F3=Exit    F12=Cancel
```

Press the Enter key to return to the Work with DDM Files display.

In addition to displaying the details of the DDM file, you can *print* the detail information by typing a 6 (Print details) in the *Option* column.

You can also print a list of the DDM files by pressing F9 (Print list).

To *delete* a file or files, type a 4 (Delete) in the *Option* column next to the files you want to delete or in the top list entry and specify the file you want to delete.

```
                           Work with DDM Files

  Position to  . . . . . _____________________

  Type options, press Enter.
    1=Create DDM file   2=Change DDM file   4=Delete   5=Display details
    6=Print details

                                                                  Remote
  Option   Local File              Remote File                  Location
    _      _____________________
    _      WILSON/A                  A                            S36
    4      WILSON/TEST               TESTFILE.TESTLIB             S38









                                                                  Bottom
  F3=Exit   F5=Refresh   F9=Print list   F12=Cancel
```

Press the Enter key. You are shown the Confirm Delete of Files display.

```
                      Confirm Delete of Files

Press Enter to confirm your choices for 4=Delete.
Press F12 to return to change your choices.
                                                                  Remote
Option   Local File             Remote File                      Location
  4      WILSON/TEST            TESTFILE.TESTLIB                 S38

                                                                    Bottom
F12=Cancel
```

Choose one of the actions on the display and then press the Enter key. You return to the Work with DDM Files display.

Related reference

"CHGDDMF (Change DDM File) command" on page 67
The Change DDM File (CHGDDMF) command changes one or more of the attributes of a DDM file on the local (source) server.

"CRTDDMF (Create DDM File) command" on page 67
The Create DDM File (CRTDDMF) command creates a DDM file on the local (source) server.

"DSPDDMF (Display DDM Files) command" on page 68
The Display DDM Files (DSPDDMF) command displays the details of a DDM file.

DDM-related CL command considerations

This topic collection describes DDM-related specifics about iSeries CL commands when they are used with DDM files. These topics discuss running the commands on the source server and do not discuss them being submitted to run on the target server by the Submit Remote Command (SBMRMTCMD) command.

Note: You see message CPF9810 if the following items are true about a DDM file:

- The file is created into library QTEMP.
- The file is used by a CL command (such as CPYF).
- A remote file and library was specified in the CL command and the library does not exist on the remote server.

Message CPF9810 indicates that the QTEMP library was not found. However, the library that was not found is the remote library that was specified in the DDM file.

Related concepts

Control language

Related reference

"DDM-related CL command lists" on page 98
The control language (CL) commands that have a specific relationship with DDM are grouped in charts in these topics to show the command functions that are available with DDM, those having common limitations when used with DDM, and those that cannot be used with DDM.

File management handling of DDM files

Because of the way data management handles DDM files, you must be careful when specifying a member name on commands. If a member name is specified, data management first searches for a local database file containing the member specified before looking for a DDM file.

For example, assume the following items:

- DDM file CUST021 is in library NYCLIB.
- Database file CUST021 is in library CUBSLIB.

NYCLIB is listed before CUBSLIB in the user's library list. CUBSLIB/CUST021 contains member NO1. The remote file pointed to by the DDM file contains member NO1. If the following override is used on an Override with Database File (OVRDBF) command:

```
OVRDBF FILE(CUST021) MBR(NO1)
```

Data management finds the database file CUBSLIB/CUST021 instead of the DDM file NYCLIB/CUST021.

To avoid this, you can do one of the following things:

- Qualify the TOFILE on the override:

  ```
  OVRDBF FILE(CUST021) TOFILE(NYCLIB/CUST021) MBR(NO1)
  ```

- Remove the library containing the database file from the library list:

  ```
  RMVLIBLE LIB(CUBSLIB)
  ```

- Remove the override and change the remote file name in the DDM file to contain the member name:

  ```
  CHGDDMF FILE(NYCLIB/CUST021)
     RMTFILE(*NONSTD 'XYZ/CUSTMAST(NO1)')
  ```

ALCOBJ (Allocate Object) command

When the name of a DDM file is specified on the Allocate Object (ALCOBJ) command on the source server, the command allocates the DDM file on the source server and its associated file or file member on a target server.

The command places locks on both the DDM file and the remote file in each pair. (These files are locked on both servers to ensure that they are not changed or deleted while the files or members are locked.) One or more pairs of files (DDM files on the source server and remote files on one or more target servers) can be allocated at the same time.

Each DDM file is always locked with a shared-read (*SHRRD) lock. Shared-read is used for the DDM files regardless of the lock types that might have been specified on the command to lock other local files at the same time.

The lock placed on the *remote file* depends on the type of target server:

- When the target is an iSeries server or a System/38, the resulting locks on the remote file are the same as if the file is a local database file. That is, the iSeries or the System/38 remote file is also locked with a shared-read lock, and the member (the one specified, or the first one) is locked with the lock type specified on the command.
- When the target is *not* an iSeries server or a System/38, the remote file is locked with the specified lock type, except that some non-iSeries target servers might use a stronger lock than was specified on the command. If an ALCOBJ command specifies multiple DDM files, and one or more are on non-iSeries target servers, those remote files are locked with the lock type specified on the command. If a member name is specified for a remote server that does not support members, the lock request is rejected with an error message, unless the member name is the same as the DDM file name.

Related reference

"Allocate Object (ALCOBJ) and Deallocate Object (DLCOBJ) commands" on page 120
The ALCOBJ command locks DDM files on the source server and the associated remote files on the target servers.

Member names and iSeries target servers on the ALCOBJ command:

If a member name is specified with the DDM file name on an ALCOBJ command, the member (in the remote file) is locked with the lock type specified on the command.

If a member name is also specified in the DDM file itself, the member names on both commands (ALCOBJ and CRTDDMF) must be the same. If they are different, the lock request is rejected and an error message is sent to the user of the program. The remote file containing the member is locked with a shared-read lock regardless of the lock type specified for the member.

If no member name is specified when a DDM file name is specified on an ALCOBJ command for a remote file on an iSeries server or a System/38, *FIRST is the default, and the target server attempts to locate and lock the first member in the remote file, the same as if it had been specified by name. If a remote file has no members, the lock request is rejected with an error message.

Lock multiple DDM files with the ALCOBJ command:

One ALCOBJ command can be used to specify multiple DDM files that are associated with remote files located on multiple target servers. If it is not possible to lock all the files on all the servers, none are locked.

ALCOBJ command completion time with DDM:

When DDM-related files are being allocated, a longer time will be required for the command to complete because of the additional time required for communications to occur between the source and target servers.

You should not, however, increase the wait time specified in the WAIT parameter on the Allocate Object (ALCOBJ) command; communications time and the WAIT parameter value have no relationship with each other.

Note: If the DLTF command is used to delete the remote file without first releasing (using the DLCOBJ command) the locks obtained by the ALCOBJ command, the DDM conversation is not reclaimed until the source job has ended.

CHGJOB (Change Job) command

The Change Job (CHGJOB) command can be used to change the DDMCNV parameter, which controls whether Advanced Program-to-Program Communication (APPC) or iSeries Access Family conversations allocated for DDM use are to be kept active or automatically dropped when they are not in use by a job. The new value goes into effect immediately for the specified job.

To display the current value of the DDMCNV job attribute, use the Work with Job (WRKJOB) command.

Related reference

"DDMCNV parameter considerations" on page 97
The DDMCNV parameter is a job-related parameter that controls whether Advanced Program-to-Program Communication (APPC) or iSeries conversations in the job that is allocated for DDM use (that is, DDM conversations) are to be automatically dropped or kept active for the source job.

"WRKJOB (Work with Job) command" on page 96
The Work with Job (WRKJOB) command can be used to display two DDM-related items.

CHGLF (Change Logical File) command

The Change Logical File (CHGLF) command can be used to change files on the source and target servers through the SYSTEM parameter.

Consider the following items when using the SYSTEM parameter values:

- When you specify *LCL, the logical file is changed on the local server.
- When you specify *RMT, the logical file is changed on the remote server. You must specify a DDM file on the FILE parameter.
- When you specify *FILETYPE, a remote file is changed if a DDM file has been specified on the FILE parameter. If a DDM file has not been specified, a local logical file is changed.

Consider the following items when using this command with DDM:

- The FILE parameter is the name of the DDM file that represents the remote logical file being changed. The remote file specified on the DDM file is the logical file that is changed on the remote server (which is also specified in the DDM file).
- For a target server other than an iSeries server:
  - All parameters except TEXT are ignored.
  - It is not verified that the remote file is a logical file.

CHGPF (Change Physical File) command

The Change Physical File (CHGPF) command can be used to change files on the source and target systems through the SYSTEM parameter.

Consider the following items when using the SYSTEM parameter values:

- When you specify *LCL, the physical file is changed on the local system.
- When you specify *RMT, the physical file is changed on the remote system. You must specify a DDM file on the FILE parameter.
- When you specify *FILETYPE, if a DDM file has been specified on the FILE parameter, a remote file is changed. If a DDM file has not been specified, a local physical file is changed.

Consider the following items when using this command with DDM:

- The FILE parameter is the name of the DDM file that represents the remote physical file being changed. The remote file specified in the DDM file is the physical file that is changed on the remote system (which is also specified in the DDM file).
- For a target server other than an iSeries server:
  - All parameters except EXPDATE, SIZE, and TEXT are ignored.
  - It is not verified that the remote file is a physical file.

CHGSRCPF (Change Source Physical File) command

The Change Source Physical File (CHGSRCPF) command can be used to change files on the source and target servers through the SYSTEM parameter.

Consider the following items when using the SYSTEM parameter values:

- When you specify *LCL, the source physical file is changed on the local server.
- When you specify *RMT, the source physical file is changed on the remote server. You must specify a DDM file on the FILE parameter.
- When you specify *FILETYPE, if a DDM file has been specified on the FILE parameter, a remote file is changed. If a DDM file has not been specified, a local source physical file is changed.

Consider the following items when using this command with DDM:

- The FILE parameter is the name of the DDM file that represents the remote source physical file being changed. The remote file specified in the DDM file is the source physical file that is changed on the remote server (which is also specified in the DDM file).
- The CCSID parameter is ignored on a target System/38 server.
- For a target server other than an iSeries server, the CHGSRCPF command cannot be used to change files.

CLRPFM (Clear Physical File Member) command

The Clear Physical File Member (CLRPFM) command can be used with DDM to clear all the records either from a physical file member on a target iSeries server or from a file on a non-iSeries target server. The command works the same way as it does for local files (clearing all data records and deleted records).

Copy commands with DDM

This topic describes the DDM implications of these CL commands.

- Copy File (CPYF)
- Copy from Query File (CPYFRMQRYF)
- Copy from Tape (CPYFRMTAP)
- Copy Source File (CPYSRCF)
- Copy to Tape (CPYTOTAP)

These commands can be used to copy data or source between files on local and remote servers. You specify with these commands which file to copy from and which file to copy to. The following table shows you what database and device files can be copied between local and remote servers.

Table 6. Copy database and device files

From file	To file
Local or remote database files	Local or remote database files
Local or remote database files	Local device files
Local device files	Local or remote database files

A DDM file is considered a device file that refers to a remote database file. Consider the following items when using these copy commands with DDM:

- DDM conversations are not reclaimed for a job when a copy command produces an error.

  Note: In releases before Version 3 Release 2, copy errors caused the Reclaim Resources (RCLRSC) command to be run, which also ran the Reclaim Distributed Data Management Conversations (RCLDDMCNV) command. The RCLRSC command is still run, but it no longer runs the RCLDDMCNV command when a copy error occurs. The DDM conversations will remain unless an explicit RCLDDMCNV is specified following the copy command with the error.
- If you specify a DDM file and a local file on the CPYF or CPYSRCF command, the server does not verify that the remote and local files are not the same file on the source server. If one DDM file is specified, a user can potentially copy to and from the same file.
- A DDM file can be specified on the FROMFILE and the TOFILE parameters for the CPYF and CPYSRCF commands.

  Note: For the Copy from Query File (CPYFRMQRYF), and Copy from Tape (CPYFRMTAP) commands, a DDM file name can be specified only on the TOFILE parameter; for the Copy to Tape (CPYTOTAP) command, a DDM file name can be specified only on the FROMFILE parameter.
- If the target server is *not* an iSeries server or a System/38:

  - When a file on the local iSeries server is copied to a remote file (or vice versa), FMTOPT(*NOCHK) is usually required.
  - When a *source* file on the local iSeries server is copied to a remote file (or vice versa), FMTOPT(*CVTSRC) *must* be specified.
- If data is copied to a target System/36 file that has alternative indexes built over it, MBROPT(*REPLACE) cannot be specified. In this case, the copy command attempts to clear the remote file, but it fails because of the alternative indexes.
- When an iSeries file that can contain deleted records is copied to one that cannot contain deleted records, you must specify COMPRESS(*YES), or an error message is sent and the job ends.
- If the remote file name on a DDM file specifies a member name, the member name specified for that file on the copy command must be the same as the member name on the remote file name on the DDM file. In addition, the Override Database File (OVRDBF) command cannot specify a member name that is different from the member name on the remote file name on the DDM file.
- If a DDM file does not specify a member name and if the OVRDBF command specifies a member name for the file, the copy command uses the member name specified on the OVRDBF command.

If the TOFILE parameter is a DDM file that refers to a file that does not exist, CPYF creates the file if CRTFILE(*YES) is specified. Listed here are special considerations for remote files created with the CPYF or CPYFRMQRYF commands:

- If the target system is an iSeries server or a System/38, the user profile for the target DDM job must be authorized to the CRTPF command on the target server.
- If the target server is a server other than an iSeries server, the file specified by the FROMFILE parameter cannot have any file or field CCSIDs other than *HEX or the CCSID of the source job.
- For the CPYF command, if the target server is a system other than an iSeries server, the FROMFILE parameter cannot be a source file.
- If the target server is a System/38, the TOMBR parameter must be the same as the remote file's name or *FIRST for the copy to be successful. The copy creates a member with the same name as the remote file's name.
- If the target server is other than a System/38 or iSeries server, for the copy to be successful, the TOMBR parameter must be *FIRST or specify the DDM file name. For DDM access to the remote file, the file appears to have a member with the same name as the DDM file.
- For an iSeries target server, the TOFILE parameter has all the attributes of the FROMFILE parameter.
- For target systems that are other than iSeries servers, those attributes on the CRTPF command that are ignored are also ignored when the copy command creates the file.
- If the target server is a System/38 and the FROMFILE parameter is a direct file that does not allow deleted records, an attempt is made to copy the records after the last record for the file at its maximum size. The system operator on the System/38 tells the server to either add the records or cancel the copy.
- The CPYF or CPYFRMQRYF command with CRTFILE(*YES) creates a file on the target server with a size description that is only as large as the target server allows.
- For all copies, if the number of records being copied exceeds the maximum allowed by the to-file, the copy function ends when the maximum is reached.
- For copy commands executed on Version 2 Release 3 or earlier systems that reference a Version 3 Release 1 remote file having a constraint relationship, the ERRLVL parameter will not work for constraint relationship violations. The copy ends regardless of the ERRLVL specified.
- The copy commands allow copying from and to DDM files that reference remote distributed files.

CRTDTAARA (Create Data Area) command

The Create Data Area (CRTDTAARA) command creates a data area and stores it in a specified library. It also specifies the attributes of the data. The data area can be optionally initialized to a specific value.

You can create a DDM data area by specifying *DDM on the TYPE parameter. The DDM data area is used as a reference data area by programs to access data areas located on a remote (target) server in the

DDM network. Programs on the local (source) server reference a remote data area by the DDM data area's name, not by the remote data area's name. (The DDM data area name can be the same as the remote data area name.)

The DDM data area (on the source server) contains the name of the remote data area and the name of the remote (target) server on which the remote data area is located.

The DDM data area can be used with the Retrieve Data Area (RTVDTAARA) command and the Change Data Area (CHGDTAARA) command to retrieve and update data areas on remote servers. A DDM data area can also be used with the Retrieve Data Area (QWCRDTAA) API.

Consider the following items when using this command with DDM:

- The RMTDTAARA parameter is the name of the remote data area on the target server. The data area does not need to exist when the DDM data area is created.
- The RMTLOCNAME parameter is the name of the remote location that is used with this object. Multiple DDM data areas can use the same remote location for the target system. RMTLOCNAME must point to a target server that is an iSeries running at a release of i5/OS that supports remote data areas. The possible values for RMTLOCNAME include:
  - `remote-location-name`: Specifies the name of the remote location that is associated with the target system. The remote location, which is used in accessing the target system, does not need to exist when the DDM data area is created, but it must exist when the DDM data area is accessed.
  - *RDB: The remote location information for the relational database entry specified in the relational database (RDB) parameter is used to determine the remote system.
- The DEV parameter is the name of the APPC device description on the source server that is used with this DDM data area. The device description does not need to exist when the DDM data area is created.
- The LCLLOCNAME parameter is the local location name.
- The MODE parameter is the mode name that is used with the remote location name to communicate with the target server.
- The RMTNETID parameter is the remote network ID in which the remote location resides that is used to communicate with the target server.

Consider the following restrictions when using this command with DDM:

- You cannot create a DDM data area using the names *LDA, *GDA, or *PDA.
- You cannot create a data area remotely. This function can be done remotely by using the Submit Remote Command (SBMRMTCMD) command.
- You can remotely display data areas by using the SBMRMTCMD command.
- You can display the contents of remote data areas by using the Display Data Area (DSPDTAARA) command; specify *RMT on the SYSTEM parameter. The data in the data area is displayed in the same format as that used for local data areas, with the exception of the TEXT field, which is the text description provided when the DDM data area was created. If you specify *LCL on the SYSTEM parameter for a DDM data area, the output looks similar to the following display:

```
Data area . . . . . :     DDMDTAARA
Library . . . . . :        DDMLIB
Type    . . . . . :   *DDM
Length    . . . . . :   62
Text    . . . . . :   'This is a DDM data area'

          Value
Offset        *...+....1....+....2....+....3....+....4....+....5
   0         '*LOC      *NETATR SYSTEMA *LOC    *LOC    LCLDTAAR'
  50         'A LCLLIB    '
```

Use the following chart to interpret the values:

Table 7. Offset values

Offset	DDMDTAARA fields
1-10	DEV
11-18	MODE
19-26	RMTLOCNAME
27-34	LCLLOCNAME
35-42	RMTNETID
43-52	RMTDTAARA (name)
53-62	RMTDTAARA (library)

Related concepts

Control language

CRTDTAQ (Create Data Queue) command

The Create Data Queue (CRTDTAQ) command creates a data queue and stores it in a specified library. Data queues are used to communicate and store data used by several programs either within a job or between jobs. Multiple jobs can send or receive data from a single queue.

The CRTDTAQ command can optionally create a distributed data management (DDM) data queue. This is done by specifying *DDM on the TYPE parameter. The DDM data queue is used as a reference data queue by programs to access data queues located on a remote (target) server in the DDM network. Programs on the local (source) server reference a remote data queue by the DDM data queue's name, not by the remote data queue's name. (The DDM data queue name, however, can be the same as the remote data queue name.)

The DDM data queue (on the source server) contains the name of the remote data queue and the name of the remote (target) server on which the remote data queue is located.

Consider the following items when using this command with DDM:

- The TYPE parameter specifies the type of data queue to be created. A standard data queue or a DDM data queue can be created.
- The RMTDTAQ parameter is the name of the remote data queue on the target system. The data queue does not need to exist when the DDM data queue is created.
- The RMTLOCNAME parameter is the name of the remote location that is used with this object. Multiple DDM data areas can use the same remote location for the target system. RMTLOCNAME must point to a target server that is an iSeries running at a release of i5/OS that supports remote data areas. The possible values for RMTLOCNAME include:
  - `remote-location-name`: Specifies the name of the remote location that is associated with the target system. The remote location, which is used in accessing the target system, does not need to exist when the DDM data area is created, but it must exist when the DDM data area is accessed.
  - *RDB: The remote location information for the relational database entry specified in the relational database (RDB) parameter is used to determine the remote system.
- The DEV parameter is the name of the APPC device description on the source system that is used with this DDM data queue. The device description does not need to exist when the DDM data queue is created.
- The LCLLOCNAME parameter is the local location name.
- The MODE parameter is the mode name that is used with the remote location name to communicate with the target system.
- The RMTNETID parameter is the remote network ID in which the remote location resides that is used to communicate with the target system.

Consider the following restrictions when using this command with DDM:

- Only the API interface for data queues is supported when using DDM data queues. The following APIs are supported:
  - Send to Data Queue (QSNDDTAQ)
  - Receive from Data Queue (QRCVDTAQ)
  - Clear Data Queue (QCLRDTAQ)

  The Retrieve Data Queue Description (QMHQRDQD) and Retrieve Data Queue Messages (QMHRDQM) APIs are not supported for DDM data queues.

  When using the *ASYNC parameter on the Send Data Queue API, messages resulting from errors encountered when accessing the remote data queue are placed in the target server's job log, and a DDM protocol error (CPF9173 - `Error detected in DDM data stream by target server`) is posted in the source system's job log. Look in the target server's job log for the cause of the error and correct the problem before using the remote data queue. Attempts to access the remote data queue after you receive this error message without first correcting the problem will produce unpredictable results.
- You cannot create a data queue remotely. This function can be done remotely by using the Submit Remote Command (SBMRMTCMD) command.

Related concepts

Control language

Application programming interfaces

CRTLF (Create Logical File) command

The Create Logical File (CRTLF) command can be used to create files on the source and target servers through the SYSTEM parameter.

Consider the following items when using the SYSTEM parameter values:

- When you specify *LCL, the file is created on the local server.
- When you specify *RMT, the file is created on the remote server. You must specify a DDM file on the FILE parameter.
- When you specify *FILETYPE, if a DDM file has been specified on the FILE parameter, a remote file is created. If a DDM file has not been specified, a local file is created.

Consider the following items when using this command with DDM:

- The parameter FILE is the name of the DDM file that represents the remote logical file being created. The remote file specified in the DDM file is the logical file that is created on the remote server (which is also specified in the DDM file).
- The OPTION and GENLVL parameters have no effect on the remote command sent.
- The files specified on the PFILE or JFILE keywords in the DDS for the logical file must be at the same server location as the logical file being created.
- If *JOB is specified as the value of a parameter or is in the data description specification (DDS) for that file, the attribute of that source job is used for file and field attributes. The attribute of the source job is also used when the default for a file or field attribute is the job attribute.
- For a target server other than an iSeries server:
  - The format name is ignored.
  - Only the value of *ALL is supported for the DTAMBRS parameter.
  - These parameters are ignored:
    - AUT
    - FRCRATIO
    - FRCACCPTH
    - LVLCHK
    - MAINT

- MBR
- RECOVER
- SHARE
- UNIT
- WAITFILE
- WAITRCD

Note: For System/38 targets, the SBMRMTCMD command can be used to change these attributes.

– Only the value of *NONE is supported for the FMTSLR parameter.
– FILETYPE must be *DATA.
– If a member name is specified, it must match the DDM file name.

- For an iSeries target server:
  – All parameters of the CRTLF command are supported with one restriction: authorization lists are not allowed for the AUT (public authority) parameter. DDM cannot guarantee the existence of the authorization list on the target server or that the same user IDs are in the list if it does exist. The public authority is changed to *EXCLUDE when you use an authorization list as a value for the AUT parameter of the CRTLF command.
  – The file names specified in the DTAMBRS parameter must be the names of the DDM files that represent the remote based-on physical files. If a member name was specified as part of the remote file name of the DDM file, then only that member name can be specified. The member names must be the actual remote file member names.

CRTPF (Create Physical File) command

The Create Physical File (CRTPF) command can be used to create files on the source and target servers through the SYSTEM parameter.

Consider the following items when using the SYSTEM parameter values:

- When you specify *LCL, the file is created on the local server.
- When you specify *RMT, the file is created on the remote server. You must specify a DDM file on the FILE parameter.
- When you specify *FILETYPE, if a DDM file has been specified on the FILE parameter, a remote file is created. If a DDM file has not been specified, a local file is created.

Consider the following items when using this command with DDM:

- The FILE parameter is the name of the DDM file that represents the remote file being created. The remote file specified in the DDM file is the file that is created on the remote server (which is also specified in the DDM file).
- The OPTION and GENLVL parameters create the same results as for local processing. These parameters have no effect on the remote command sent.
- If *JOB is specified as the value of a parameter or is in the data description specification (DDS) for that file, the attribute of that source job is used for file and field attributes. The attribute of the source job is also used when the default for a file or field attribute is the job attribute.
- For a target server other than an iSeries server:
  – The format name is ignored.
  – These parameters are ignored:
    - AUT
    - CONTIG
    - DLTPCT
    - FRCRATIO
    - FRCACCPTH

- LVLCHK
- MAINT
- MAXMBRS2
- MBR
- RECOVER
- REUSEDLT
- SHARE
- UNIT
- WAITFILE
- WAITRCD

Note: For System/38 targets, the SBMRMTCMD command can be used to change these attributes.

– FILETYPE must be *DATA.
– All other parameters are supported.
– If a member name is specified, it must match the DDM file name.
– The only CCSID values that are supported are:
  - *HEX
  - 65535
  - *JOB
  - Process CCSID of the source job

  The file is not created if any other CCSID value is specified.
– When the DDS keyword VARLEN is used, DDM tries to create a variable-length record file on the target server. There are some specific rules for this keyword.

• On an iSeries target server, all parameters of the CRTPF command are supported with one restriction: authorization lists are not allowed for the AUT (public authority) parameter. DDM cannot guarantee the existence of the authorization list on the target server or that the same user IDs are in the list if it does exist. The public authority is changed to *EXCLUDE when you use an authorization list as a value for the AUT parameter of the CRTPF command.

Related reference

"DDM-related DDS keywords and information" on page 109
The information about DDS keywords that relates specifically to DDM is provided in this topic.

CRTSRCPF (Create Source Physical File) command

The Create Source Physical File (CRTSRCPF) command can be used to create files on the iSeries source and target servers through the SYSTEM parameter.

Consider the following items when using the SYSTEM parameter values:

• When you specify *LCL, the file is created on the local server.
• When you specify *RMT, the file is created on the remote server. You must specify a DDM file on the FILE parameter.
• When you specify *FILETYPE, if a DDM file has been specified on the FILE parameter, a remote file is created. If a DDM file has not been specified, a local file is created.

Consider the following items when using this command with DDM:

• The FILE parameter is the name of the DDM file that represents the remote file being created. The remote file specified in the DDM file is the file that is created on the remote server (which is also specified in the DDM file).
• The OPTION and GENLVL parameters create the same results as for local processing. These parameters have no effect on the remote command sent.

- If *JOB is specified as the value of a parameter or is in the data description specification (DDS) for that file, the attribute of that source job is used for file and field attributes. The attribute of the source job is also used when the default for a file or field attribute is the job attribute.

All parameters of the CRTSRCPF command are supported with one restriction: authorization lists are not allowed for the AUT (public authority) parameter. DDM cannot guarantee the existence of the authorization list on the target server or that the same user IDs are in the list if it does exist. The public authority is changed to *EXCLUDE when you use an authorization list as a value for the AUT parameter of the CRTSRCPF command.

DLCOBJ (Deallocate Object) command

When the name of a DDM file is specified on the Deallocate Object (DLCOBJ) command on the source server, the command deallocates the DDM file on the source server and its associated file or file member on a target server.

The command releases the locks that were placed on the paired files on both the source and target servers by the Allocate Object (ALCOBJ) command. One or more pairs of files (DDM files on the source server and remote files on one or more target servers) can be deallocated at the same time.

Related reference

"Allocate Object (ALCOBJ) and Deallocate Object (DLCOBJ) commands" on page 120
The ALCOBJ command locks DDM files on the source server and the associated remote files on the target servers.

Member names and iSeries target servers on the DLCOBJ command:

All of the information previously discussed in the ALCOBJ command description regarding member names applies to the DLCOBJ command as well.

Refer to the ALCOBJ command description for the details.

Unlock multiple DDM files on the DLCOBJ command:

One DLCOBJ command can be used to specify multiple DDM files that are associated with remote files that might be located on multiple target servers. In most cases, the command attempts to release as many of the specified locks as possible.

For example:

- If one of the DDM files specified on the DLCOBJ command refers to a remote file that is not a database file, that lock is not released; but the specified locks on the remote files associated with the other DDM files specified are released if, of course, they are valid.
- If a user tries to release a lock that he did not place on a file by a previous ALCOBJ command, that part of the request is rejected and an informational message is returned to the user.

DLTF (Delete File) command

The Delete File (DLTF) command can be used to delete files on the source and target servers.

The following items should be considered when using the SYSTEM parameter values:

- When you specify *LCL, only local files are deleted. This might include DDM files.
- When you specify *RMT, the file is deleted on the remote server. You must specify a DDM file on the FILE parameter. If a generic name is specified, the remote files corresponding to any DDM files matching the generic name are deleted. (The local DDM files are not deleted.)
- When you specify *FILETYPE, if a DDM file has been specified, the remote file is deleted. If a DDM file has not been specified, the local file is deleted. When you specify generic names, local non-DDM files are deleted first. Remote files for any DDM files matching the generic name are then deleted. Local DDM files are not deleted.

Notes:

1. Structured Query Language/400 (SQL/400) DROP TABLE and DROP VIEW statements work only on local files.
2. If the DLTF command is used to delete the remote file without first releasing (using the DLCOBJ command) the locks obtained by the ALCOBJ command, the DDM conversation is not reclaimed until the source job has ended.

DSPFD (Display File Description) command

The Display File Description (DSPFD) command can be used to display (on the source server) the attributes of the DDM file on the source server, the remote file on the target server, or both the DDM file and the remote file. As with local files, the attributes of multiple DDM files, multiple remote files, or both can be displayed by the same command.

Note: Although this discussion mentions only one target server, the files for multiple target servers can be displayed at the same time.

The SYSTEM parameter determines which group of attributes is displayed.

- To display the attributes of DDM files, which are *local* files, the SYSTEM parameter must specify *LCL (the default). If SYSTEM(*LCL) is specified:
  - The FILEATR parameter must either specify *DDM (to display DDM file attributes only) or default to *ALL (to display all file types, including DDM files). The same kind of information is displayed for DDM files (which are on the local system) as for any other types of files on the local server.
  - If FILEATR(*DDM) is specified and the OUTFILE parameter specifies a file name, only local DDM file information is given.
- To display the attributes of remote files, the SYSTEM parameter must specify *RMT. If SYSTEM(*RMT) is specified:
  - The FILEATR parameter must specify *ALL, *PHY, or *LGL.
  - The type of information displayed for remote files depends on what type of target server the files are on. If the target is an iSeries server or a System/38, the same type of information displayed for local files on an iSeries server or a System/38 can be displayed. If the target is *not* an iSeries server or a System/38, all the information that can be obtained through that server's implementation of the DDM architecture that is compatible with the iSeries server's implementation is displayed.
- To display the attributes of both DDM and remote files, the SYSTEM parameter must specify *ALL.

Related reference

"Display DDM remote file information" on page 122
The CL commands Display File Description (DSPFD) and Display File Field Description (DSPFFD) can be used by an iSeries source server user to display the attributes of one or more DDM files on the source server, or to display the attributes of one or more remote files on a target server.

DSPFFD (Display File Field Description) command

The Display File Field Description (DSPFFD) command can be used to display the file, record format, and field attributes of a remote file. To display the remote file attributes, however, you must enter the name of the DDM file associated with the remote file, not the name of the remote file.

Note: Because the DDM file has no field attributes, the DSPFFD command cannot specify SYSTEM(*LCL) to display local DDM file information.

If *ALL or a generic file name is specified on the FILE parameter, the DSPFFD command can also display information about a group of both local files and remote files, or just a group of local files. In this case, the SYSTEM parameter determines which are displayed.

- To display the attributes of *local non-DDM* files only, the SYSTEM parameter need not be specified because *LCL is the default.

- To display the attributes of *remote* files, the SYSTEM parameter must specify *RMT. If SYSTEM(*RMT) is specified, the field and record format information displayed for remote files depends on what type of target server the files are on.
  - If the target is an iSeries server or a System/38, the same information displayed for local files on an iSeries server is displayed.
  - If the target is other than a System/38 or iSeries server:
    - Fields are F*nnnnn* or K*nnnnn* (where *nnnnn* is some number), based on whether the file is a keyed file or not.
    - The record format name is the DDM file name.

    If the remote file has a record length class of record varying or initially varying, fixed-length field descriptions are displayed.
- To display the attributes of both *local non-DDM files and remote files*, the SYSTEM parameter must specify *ALL. Only remote physical and logical files can be displayed.

Related reference

"Display DDM remote file information" on page 122
The CL commands Display File Description (DSPFD) and Display File Field Description (DSPFFD) can be used by an iSeries source server user to display the attributes of one or more DDM files on the source server, or to display the attributes of one or more remote files on a target server.

OPNQRYF (Open Query File) command

You can query remote files using the Open Query File (OPNQRYF) command, but only if the remote files are on a target iSeries server or a target System/38.

If multiple remote files are specified on one OPNQRYF command, they must all exist on the same target server and use the same remote location information.

If the target server is an iSeries server or a System/38, a query request is created and sent to the target server using the DDM file that the query refers to. If the target server is other than an iSeries server or a System/38, the query request cannot be processed and an error message is created. However, the query utility on the System/38 can be used to query remote files that are other than iSeries files.

If the target server is a System/38 and the source is an iSeries server, or if the target server is an iSeries server and the source is a System/38, OPNQRYF cannot use group-by and join functions. An error results.

Related reference

"i5/OS database query" on page 38
The database interactive query function, provided by the i5/OS licensed program, supports DDM files.

"System/38-compatible query utility (Query/38)" on page 34
The System/38-compatible query utility (Query/38) can be used with DDM to create and use interactive or batch query applications.

OVRDBF (Override with Database File) command

The Override with Database File (OVRDBF) command can be used with DDM to override (replace) a local database file named in the program with a DDM file; the DDM file causes the associated remote file to be used by the program instead of the local database file.

If a DDM file is specified on the TOFILE parameter and if other parameters are specified that change a file's attributes, the result is that the remote file actually used by the program is used with its attributes changed by the parameter values specified on the OVRDBF command.

If the target server is an iSeries server or a System/38, existing programs that use the OVRDBF command to access remote files work the same as when they access local files. All the OVRDBF parameters are processed the same on source and target iSeries servers.

If end-of-file delay (EOFDLY) is used, it is recommended to end a job with an end-of-file record because if the source job gets canceled, the target job does not get notified. The user must also end the target job.

If the target server is neither an iSeries server nor a System/38:

- The following parameters are still valid: TOFILE, POSITION, RCDFMTLCK, WAITFILE, WAITRCD, LVLCHK, EXPCHK, INHWRT, SECURE, SHARE, and SEQONLY.
  - The TOFILE parameter is always processed on the source server. When a DDM file name is specified on this parameter, the program uses the associated remote file instead of the local database file specified in the program.
  - The RCDFMTLCK parameter, if specified, is valid only if both of the following are true of the remote file used: only one type of lock condition can be requested for the remote file, and the record format name in the remote file must be the same as the name of the DDM file.
  - The WAITFILE and WAITRCD parameters have no effect on remote file processing.
- The MBR parameter causes an error if it is specified with a member name that is different than the name of the file containing the member.
- The FRCRATIO and NBRRCDS parameters, if specified, are ignored.
- The FMTSLR parameter, if specified, causes an error when the file being opened is a DDM file.
- The SEQONLY parameter causes records to be blocked on the source side. Records might be lost if the source job is canceled before a block is full.

Related reference

"Additional considerations: SBMRMTCMD command" on page 72
This topic describes additional considerations for the SBMRMTCMD command.

"Example: Access DDM remote members (iSeries server only)" on page 116
These examples show how access to a DDM file becomes an indirect reference (by using DDM) to a member of a file on a remote iSeries server. These examples are iSeries server-to-iSeries server examples.

RCLRSC (Reclaim Resources) command

The Reclaim Resources (RCLRSC) command, like the Reclaim DDM Conversations (RCLDDMCNV) command, can be used to reclaim all DDM conversations that currently have no users in the job.

This can be done even if the DDMCNV job attribute is *KEEP. The RCLRSC command, however, first attempts to close any unused files for the appropriate recursion levels, as it would for local files. This action might result in some conversations allocated to DDM being unavailable for the job. For example, if a DDM file is opened using the Open Database File (OPNDBF) command, the RCLRSC command closes the file and reclaims the conversation.

After the files are closed, any unused DDM conversations are dropped. Whether a conversation can be reclaimed is not affected by the recursion level or activation group in which the RCLRSC command is issued.

Related reference

"Control DDM conversations" on page 120
Normally, the DDM conversations associated with a source server job are kept active until one of the conditions described in this topic is met.

RNMOBJ (Rename Object) command

The Rename Object (RNMOBJ) command can be used to rename a remote file.

The following items should be considered when using the SYSTEM parameter values:

- When you specify *LCL, local objects are renamed. This might include DDM files.
- When you specify *RMT, this value applies only to OBJTYPE(*FILE). The DDM file containing the remote file to be renamed is specified on the OBJ parameter.

  The DDM file containing the new name for the remote file is specified on the NEWOBJ parameter. Both DDM files must already exist in the same library (on the source server). The two DDM files must refer to the same target servers and contain the same remote location information. Neither the two local DDM files nor the RMTFILE names in the two DDM files are changed. Specify *LCL to rename the DDM file or use the Change DDM File (CHGDDMF) command to change the RMTFILE name in a DDM file.
- When you specify *FILETYPE, this value applies only to OBJTYPE(*FILE). If the file specified in the OBJ parameter is a DDM file, the rules when specifying *RMT apply. If the file is not a DDM file, the rules when specifying *LCL apply.

When renaming remote files for iSeries and System/38 targets, if library names have been specified in the RMTFILE parameter for the two DDM files, the library names must be the same but the file names must be different.

WRKJOB (Work with Job) command

The Work with Job (WRKJOB) command can be used to display two DDM-related items.

These items include:

- The DDMCNV job attribute for the source job.
- The object lock requests (held locks and pending locks) for DDM files that are being used in the source server job. These are shown by choosing option 12 (Work with locks, if active) from the Work with Job menu.

The Job Locks display shows only the locks held for the local DDM files; locks for remote files are not shown. Also, because DDM files do not have members, none are indicated on this display nor on the Member Lock display.

An iSeries server does not display any locks for remote files; locks for the remote file, its members, or its records cannot be displayed by the source server. However, these remote locks can be displayed using pass-through.

The lock condition shown for DDM files is always shared read (*SHRRD) regardless of the lock conditions used for their associated remote files or members.

Related reference

"DDMCNV parameter considerations" on page 97
The DDMCNV parameter is a job-related parameter that controls whether Advanced Program-to-Program Communication (APPC) or iSeries conversations in the job that is allocated for DDM use (that is, DDM conversations) are to be automatically dropped or kept active for the source job.

"Work with Job (WRKJOB) and Work with Object Locks (WRKOBJLCK) commands" on page 120
For both the WRKOBJLCK command and menu option 12 (Work with locks, if active) of the WRKJOB command, only the locks held for the local DDM files are shown, *not* locks held for the remote files (or for their members).

WRKOBJLCK (Work with Object Lock) command

The Work with Object Lock (WRKOBJLCK) command can be used to display the object lock requests (held locks and pending locks) for DDM files. This command displays only the locks held for the local DDM files, not locks held for the associated remote files.

An iSeries server does not display any locks for remote files; locks for the remote file, its members, or its records cannot be displayed by the source server.

The lock condition shown for DDM files is always shared read (*SHRRD) regardless of the lock conditions used for their associated remote files or members.

Related reference

"Work with Job (WRKJOB) and Work with Object Locks (WRKOBJLCK) commands" on page 120
For both the WRKOBJLCK command and menu option 12 (Work with locks, if active) of the WRKJOB command, only the locks held for the local DDM files are shown, *not* locks held for the remote files (or for their members).

DDM-related CL parameter considerations

This topic covers parameter considerations that apply to DDM-related CL commands.

Note: The Create DDM File (CRTDDMF) command can be used to create a DDM file. The other create file commands such as CRTPF or CRTxxxF cannot be used to create a DDM file.

Related reference

"Commands not supporting DDM" on page 102
These CL commands are not supported for DDM files. However, useful results for some of them might be produced on a target iSeries server or a System/38 using DDM if they are submitted on the Submit Remote Command (SBMRMTCMD) command to run on the target server.

DDMACC parameter considerations

The DDMACC parameter controls how an iSeries server, as a target server, handles DDM requests from other servers.

The DDMACC parameter is used on the Change Network Attributes (CHGNETA), Display Network Attributes (DSPNETA), and Retrieve Network Attributes (RTVNETA) commands. The value of this server-level parameter determines whether this iSeries server can accept DDM requests from other servers.

DDMCNV parameter considerations

The DDMCNV parameter is a job-related parameter that controls whether Advanced Program-to-Program Communication (APPC) or iSeries conversations in the job that is allocated for DDM use (that is, DDM conversations) are to be automatically dropped or kept active for the source job.

The default is to keep the conversation active.

This parameter can drop a conversation when it has no active users. The conversation is unused when:

1. All the DDM files and remote files used in the conversation are closed and unlocked (deallocated).
2. No other DDM-related functions (like the Submit Remote Command (SBMRMTCMD) command or the Display File Description (DSPFD) command to access the target server) are being done.
3. No DDM-related function has been interrupted (by a break program, for example) while running.

The DDMCNV parameter values are:

***KEEP**
: Specifies that once each DDM conversation is started for the source job it is kept active at the completion of a source program request, and it waits for another request to be received from the same or another program running in the job. This is the default value.

***DROP**
: Specifies that each DDM conversation started in the source job is to be automatically dropped when both of the following items are true: no request from the source server program(s) running in the job is being processed in the conversation, and all the DDM files are closed and all objects that were allocated are now deallocated.

The DDMCNV parameter is changed by the Change Job (CHGJOB) command and is displayed by the Work with Job (WRKJOB) command. Also, if the Retrieve Job Attributes (RTVJOBA) command is used, you can get the value of this parameter and use it within a CL program.

Related concepts

"iSeries server as the source server for DDM" on page 13
When an application program or user in a source server job first refers to a DDM file, several actions occur as part of processing the request on the source server.

Related reference

"Additional considerations: SBMRMTCMD command" on page 72
This topic describes additional considerations for the SBMRMTCMD command.

"CHGJOB (Change Job) command" on page 83
The Change Job (CHGJOB) command can be used to change the DDMCNV parameter, which controls whether Advanced Program-to-Program Communication (APPC) or iSeries Access Family conversations allocated for DDM use are to be kept active or automatically dropped when they are not in use by a job. The new value goes into effect immediately for the specified job.

"WRKJOB (Work with Job) command" on page 96
The Work with Job (WRKJOB) command can be used to display two DDM-related items.

"Control DDM conversations" on page 120
Normally, the DDM conversations associated with a source server job are kept active until one of the conditions described in this topic is met.

OUTFILE parameter considerations for DDM

The OUTFILE parameter is used on such commands as the Display File Description (DSPFD), the Display File Field Description (DSPFFD), the Display Object Description (DSPOBJD), and the Create Auto Report Program (CRTRPTPGM). The parameter identifies a database file into which output data created by the command is stored.

When the name of a DDM file is specified on the OUTFILE parameter of these commands, two restrictions apply:

- The remote server must be an iSeries server or a System/38. This is necessary to ensure that the associated remote file has the proper format for the output data.
- The remote file associated with the DDM file must already exist. If the remote file does not exist, a message is returned to the user indicating that the remote file must exist before the function can be performed.

If the remote file named on the OUTFILE parameter does exist and is on an iSeries server or a System/38, the file will be checked for three conditions before it can be used as an output database file to store displayed output:

- The remote file must be a physical file.
- The remote file must not be a model output file. That is, it cannot be one of the model output files provided with the i5/OS operating system which has the required format, but no data.
- The record format name in the remote file must match the model output file record format name. (This condition requires that the remote system be an iSeries server or a System/38.)

If all of these conditions are met, the remote file member is cleared. (Output file members *must* be cleared before they can be used again.) If the remote file member does not exist, it is created and the output is stored in it.

DDM-related CL command lists

The control language (CL) commands that have a specific relationship with DDM are grouped in charts in these topics to show the command functions that are available with DDM, those having common limitations when used with DDM, and those that cannot be used with DDM.

Notes:

1. Not *all* of the CL commands on an iSeries server are shown in these topics. Only those that are intended (or recommended) by IBM for use with DDM or those specifically *not* intended for DDM use are shown. The intended use can be either for commands that are run on the source server to affect a remote file on the target server, or for commands that are submitted to the target server by using the Submit Remote Command (SBMRMTCMD) command to run there.
2. Some of these commands appear in more than one of the following charts.

The charts in these topics show:

- Commands affecting only the DDM file:

  Object-oriented commands that can be used with DDM files, but do not affect the associated remote files. The Create DDM File (CRTDDMF), Change DDM File (CHGDDMF), and Reclaim DDM Conversations (RCLDDMCNV) commands are included in this group.
- Commands affecting both the DDM file and the remote file:
  - File management commands that require that the target server be another iSeries server or a System/38. The SBMRMTCMD command is included in this group.
  - Member-related commands that can be used in some way on remote files.
  - Source file commands that can operate on source files while DDM is being used.

  These commands, normally used for processing local files, can (transparently to the programs) process remote files when one of their parameters specifies the name of a DDM file.
- Commands that cannot be used with DDM.

Many of these commands, when limited as shown in the charts, can still be submitted by the SBMRMTCMD command to a target server (an iSeries server or a System/38 only) to run, but it might not be useful to do so.

Related concepts

"DDM-related CL command considerations" on page 81
This topic collection describes DDM-related specifics about iSeries CL commands when they are used with DDM files. These topics discuss running the commands on the source server and do not discuss them being submitted to run on the target server by the Submit Remote Command (SBMRMTCMD) command.

Object-oriented commands with DDM

The DDM file object on the source iSeries server can be accessed by these object-oriented CL commands. These commands work with DDM files as they normally do with any other files on the local server.

Some of these commands can operate on more than one object, and one or more of them might be DDM files if, for example, a generic file name is specified.

Except as noted in the chart, these commands have no effect on the remote file associated with the DDM file; that is, no reference is made over a communications line to the target server when one of these commands specifies a DDM file.

However, if you do want one of these commands to operate on a remote file (instead of the DDM file), you can use the Submit Remote Command (SBMRMTCMD) command to submit the command to run on the target server, if it is an iSeries server or a System/38. The results of running the submitted command, in this case, are not sent back to the source server, except for some indication to the source server user (normally a message) about whether the function was performed successfully.

Command name	Descriptive name
CHGDDMF	**Change DDM File**
CHGLF[1,2,3,4]	**Change Logical File**
CHGOBJOWN	**Change Object Owner**

Command name	Descriptive name
CHGPF[1,2,3,4]	**Change Physical File**
CHGSRCPF[1,2,3,4]	**Change Source Physical File**
CHKOBJ	**Check Object**
CRTDDMF	**Create DDM File**
CRTDUPOBJ	**Create Duplicate Object**
CRTLF[1,2,3]	**Create Logical File**
CRTPF[1,2,3]	**Create Physical File**
CRTSRCPF[1,2,3]	**Create Source Physical File**
CRTS36CBL[6]	**Create S/36 COBOL Program**
CRTS36DSPF[7]	**Create S/36 Display File**
CRTS36MNU[7]	**Create S/36 Menu**
CRTS36MSGF[7]	**Create S/36 Message File**
CRTS36RPG[6]	**Create S/36 RPG II Program**
CRTS36RPGR[7]	**Create Console Display File**
CRTS36RPT[6]	**Create S/36 RPG II Auto Report**
DLTF[1,2,3]	**Delete File**
DMPOBJ	**Dump Object**
DMPSYSOBJ	**Dump System Object**
DSPFD[1,2,3]	**Display File Description**
DSPFFD[1,2,3]	**Display File Field Description**
DSPOBJAUT	**Display Object Authority**
DSPOBJD	**Display Object Description**
GRTOBJAUT	**Grant Object Authority**
MOVOBJ	**Move Object**
RCLDDMCNV	**Reclaim DDM Conversations**
RNMOBJ[1,2,3]	**Rename Object**
RSTLIB	**Restore Library**
RSTOBJ	**Restore Object**
RVKOBJAUT	**Revoke Object Authority**
SAVCHGOBJ	**Save Changed Object**
SAVLIB	**Save Library**
SAVOBJ	**Save Object**
WRKJOB[5]	**Work with Job**
WRKOBJLCK[5]	**Work with Object Lock**

[1] When run on the source system, this command does not refer to the remote file when SYSTEM(*LCL) is used.

[2] The remote operation is performed if SYSTEM(*RMT) is specified, or if SYSTEM(*FILETYPE) is specified and the file is a DDM file.

[3] Because DDM file names can be specified on these commands, the SBMRMTCMD command is not needed to perform these functions on a target iSeries server or a target System/38.

[4] The target must be an iSeries server at release 3.0 and above or support Level 2.0 of DDM architecture.

[5] When run on the source server, this command displays any locks on the DDM file, not on the remote file.

[6] This System/36 environment command is supported by DDM.

[7] This System/36 environment command is *not* supported by DDM.

Related concepts

Control language

Related reference

"Perform file management functions on remote servers" on page 119
i5/OS DDM supports creating, deleting, or renaming of files on a remote server.

Target iSeries-required file management commands

These CL commands can be used only when the target server is another iSeries server or System/38.

Command name	Descriptive name
ADDLFM[1]	**Add Logical File Member**
ADDPFM	**Add Physical File Member**
CHGLFM	**Change Logical File Member**
CHGPFM	**Change Physical File Member**
CPYSRCF	**Copy Source File**
INZPFM	**Initialize Physical File Member**
OPNQRYF	**Open Query File**
RGZPFM	**Reorganize Physical File Member**
RMVM	**Remove Member**
RNMM	**Rename Member**

[1] The target server must be an iSeries server.

Because DDM file names can be specified on these commands, the Submit Remote Command (SBMRMTCMD) command is not needed to perform these functions on a target iSeries server or a target System/38.

Related reference

"DDM-related CL command summary charts" on page 104
This topic shows summary charts containing most of the control language (CL) commands used with DDM.

Member-related commands with DDM

Database file operations that apply to a member can be used by DDM.

When the name of a DDM file is specified on any of the following CL commands, i5/OS DDM accesses the remote file (and member) referred to by the DDM file. However, as indicated in the chart, some of these commands are valid only when the remote file is on an iSeries server or a System/38.

Command name	Descriptive name
ADDPFM[1]	**Add Physical File Member**
ADDLFM[6]	**Add Logical File Member**
ALCOBJ	**Allocate Object**
CHGLFM[1]	**Change Logical File Member**
CHGPFM[1]	**Change Physical File Member**
CLOF	**Close File**
CLRPFM	**Clear Physical File Member**
CPYF[2]	**Copy File**
CPYFRMTAP	**Copy From Tape**
CPYSPLF	**Copy Spooled File**
CPYSRCF[1]	**Copy Source File**
CPYTOTAP	**Copy To Tape**
DCLF	**Declare File**
DLCOBJ	**Deallocate Object**

Command name	Descriptive name
DSPFD[3]	**Display File Description**
DSPFFD[3]	**Display File Field Description**
DSPPFM	**Display Physical File Member**
INZPFM[1]	**Initialize Physical File Member**
OPNDBF[4]	**Open Database File**
OPNQRYF[1]	**Open Query File**
OVRDBF[5]	**Override Database File**
POSDBF	**Position Database File**
RCVF	**Receive File**
RCVNETF	**Receive Network File**
RGZPFM[1]	**Reorganize Physical File Member**
RMVM[1]	**Remove Member**
RNMM[1]	**Rename Member**
SNDNETF	**Send Network File**

[1] The target system must be an iSeries server or a System/38.

[2] For other DDM-related considerations about this command, see Copy commands with DDM.

[3] These commands display remote file information if the SYSTEM parameter specifies *RMT or *ALL.

[4] For information about commitment control, see Commitment control support for DDM.

[5] This command does not access the remote file.

[6] The target server must be an iSeries server.

The Submit Remote Command (SBMRMTCMD) command can also be used to submit some of the commands to a target server.

The Send Network File (SNDNETF) and Receive Network File (RCVNETF) commands, whenever possible, should run on the server on which the data exists, rather than using a DDM file to access the remote file.

Related reference

"DDM-related CL command summary charts" on page 104
This topic shows summary charts containing most of the control language (CL) commands used with DDM.

"Copy commands with DDM" on page 85
This topic describes the DDM implications of these CL commands.

"Use of object distribution" on page 123
Although DDM file names can be specified on the Send Network File (SNDNETF) and Receive Network File (RCVNETF) commands, these commands should be run, whenever possible, on the server where the data actually exists. Therefore, if both servers are iSeries servers and both are part of a SNADS network, *object distribution* can be used instead of DDM to transfer the data between them.

"Commitment control support for DDM" on page 27
iSeries applications can commit or roll back transactions on remote iSeries servers.

Commands not supporting DDM

These CL commands are not supported for DDM files. However, useful results for some of them might be produced on a target iSeries server or a System/38 using DDM if they are submitted on the Submit Remote Command (SBMRMTCMD) command to run on the target server.

Command name	Descriptive name
DSNFMT	Design Format
DSPCHT	Display Chart
DSPDBR	Display Database Relations
DSPRCDLCK	Display Record Locks
MNGDEVTBL	Manage Device Table
MNGPGMTBL	Manage Program Table
MNGUSRTBL	Manage User Table
RTVQRYSRC	Retrieve Query Source
SBMFNCJOB	Submit Finance Job

Related concepts

"DDM-related CL parameter considerations" on page 97
This topic covers parameter considerations that apply to DDM-related CL commands.

Source file commands

If the target server is an iSeries server or a System/38, these CL commands can support a DDM file as a source file (on the SRCFILE parameter).

If the target server is not an iSeries server or a System/38, a DDM file name should not be specified on the SRCFILE parameter, because the remote file is neither an iSeries server nor a System/38 source file.

These commands can also be affected by file overrides (by using the Override with Database File [OVRDBF] command).

Note: These commands cannot run on the source server to create a file on any target server; they can, however, be submitted to run on the target server using the Submit Remote Command (SBMRMTCMD) command.

Command name	Descriptive name
CRTBASPGM	Create BASIC Program
CRTBSCF[1]	Create BSC File
CRTCBLPGM	Create COBOL Program
CRTCLPGM	Create CL Program
CRTCMD	Create Command
CRTCMNF[1]	Create Communications File
CRTCPGM	Create C Program
CRTDSPF	Create Display File
CRTICFF	Create Intersystem Communications Function File[1]
CRTMXDF[2]	Create Mixed File
CRTPLIPGM	Create PL/I Program
CRTPRTF	Create Printer File
CRTPRTIMG[2]	Create Print Image
CRTRPGPGM	Create RPG Program
CRTRPTPGM	Create Auto Report Program
CRTTBL	Create Table
FMTDTA	Format Data

Command name	Descriptive name
STRBAS	**Start BASIC**
STRBASPRC	**Start BASIC Procedure**

[1] CRTICFF is valid on an iSeries server. CRTCMNF, CRTBSCF, and CRTMXDF commands are valid either on System/38 or System/38 environment on an iSeries server.

[2] If used with the SBMRMTCMD command, the target must be a System/38.

Related reference

"Types of files supported by i5/OS DDM" on page 111
i5/OS DDM supports all iSeries file types when the target server is another iSeries server.

DDM-related CL command summary charts

This topic shows summary charts containing most of the control language (CL) commands used with DDM.

Use these commands to determine the DDM job environment to perform remote file processing (by specifying a DDM file name on a file-related parameter of a CL command), or to perform other actions on a remote server by submitting a CL command to the target server on the Submit Remote Command (SBMRMTCMD) command.

The charts show which commands:

- Are file-related (that operate on file objects)
- Are object-related (that operate on objects other than files, in addition to file objects)
- Can be performed on the source side or on the target side
- Can be affected by file overrides by using the Override with Database File (OVRDBF) command
- Are allowed, and have a useful purpose, to be submitted to a target iSeries server to run (by using the SBMRMTCMD command), rather than running on the source server

Notes are included in the charts that can be helpful to the DDM user.

The following items describe the kinds of information provided in these charts:

- The first column lists all the CL commands that can be used by DDM: (a) to operate on a remote file identified in a DDM file, or (b) to be submitted on a SBMRMTCMD command using a DDM file.
- In the second column, an F means the command is file related, an O means it is related to i5/OS objects other than files, and a blank means neither of these.
- In the third column, an S means the command operates on objects on the source side, and a T means it operates on objects on the target side. For example, with the create commands that create a file or program using a DDM file as a source file, the T indicates that a source file on the target server is used for the creation; the command runs on the source server and creates a file or program on the source server, but uses a source file on the target server to do it.

  If neither S nor T is shown, the name of a DDM file should *not* be specified on the command; the command should not run on the *source* server as a DDM function. However, the command might be useful when submitted on the SBMRMTCMD command to run on the *target* server (see the last column).
- In the last two columns, an X indicates that the command is valid and useful when used with the command indicated at the top (OVRDBF or SBMRMTCMD) of the column. A blank indicates that the command is not valid.

Generally, when the target server is an iSeries server or a System/38, any CL command that can be used in either a batch job or batch program can be specified on the SBMRMTCMD command. If a command has a value of *BPGM *and* *EXEC specified for the ALLOW attribute, which you can display by using the

Display Command (DSPCMD) command, that command *can* be submitted by the SBMRMTCMD command. (The SBMRMTCMD command uses the QCAEXEC server program to run the submitted commands on the target server.)

Notes:

1. The SBMRMTCMD command can be used to send commands to an iSeries, System/38, or any other target server that supports the submit remote command function. The command submitted must be in the syntax of the target server.
2. Although most of the commands listed in this chart can be submitted to a remote server with the SBMRMTCMD command, several can just as easily be run on the source server specifying a DDM file name.
3. IBM grants you a nonexclusive copyright license to use all programming code examples from which you can generate similar function tailored to your own specific needs.
4. All sample code is provided by IBM for illustrative purposes only. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.
5. All programs contained herein are provided to you "AS IS" without any warranties of any kind. The implied warranties of non-infringement, merchantability and fitness for a particular purpose are expressly disclaimed.

Table 8. DDM-related CL commands

Command name	Related to file, object or both	Affects objects on source, target or both	OVRDBF command	SBMRMTCMD command[1]
ADDLFM				
ADDPFM	F	T[2]		X
ALCOBJ	F	T[3]		X
CHGDFUDEF	F O	S T		X
CHGDTA		T		X
		T		
CHGJOB				
CHGLF	F	S T		X
CHGLFM	F	T[3]		X
CHGNETA				X
CHGOBJOWN	F O	S		X
				X
CHGPF				
CHGPFM	F	S T		X
CHGQRYDEF	F	T[3]		X
CHGSRCPF		T		
CHKOBJ	F	S T		X
	F O	S		X
CLOF				
CLRPFM	F	T	X	X
COMMIT	F	T		X
CPYF	F	S T		X[11]
CPYFRMDKT	F	S T	X	X
CPYFRMQRYF	F	S T	X	X[4]
CPYFRMTAP	F	S T	X	X
	F	S T	X	X[4]
CPYSPLF				
CPYSRCF	F	T	X	X
CPYTODKT	F	S T	X	X
CPYTOTAP	F	S T	X	X[4]
CRTBASPGM	F	S T		X[4]
		T		X

Table 8. DDM-related CL commands (continued)

Command name	Related to file, object or both	Affects objects on source, target or both	OVRDBF command	SBMRMTCMD command[1]
CRTCBLPGM				
CRTCLPGM		T		X
CRTCMD		T		X
		T		X
CRTDFUAPP				
CRTDFUDEF		T		X
CRTDSPF		T		
CRTDUPOBJ	F	T		X
CRTICFF	O	S	X	X
	F	T		X
CRTLF				
CRTPF	F	S T		X
CRTPLIPGM	F	S T	X	X
CRTPRTF		T		X
CRTPRTIMG	F	T		X
		T		X
CRTQRYAPP				
CRTQRYDEF		T		X
CRTRPGPGM		T		
CRTRPTPGM		T		X
CRTSRCPF		T		X
	F	S T	X	X
CRTTBL				
DCLF		T		X
DLCOBJ	F	T		
DLTDFUAPP	F O	S T		X
DLTF				X
	F	S T		X
DLTQRYAPP				
DMPOBJ				X
DMPSYSOBJ	F O	S		X[5]
DSNDFUAPP	O	S		X[5]
DSNQRYAPP		T		
		T		
DSPDTA				
DSPFD		T		
DSPFFD	F	S T		X[5]
DSPNETA	F	S T		X[5]
DSPOBJAUT				X
	F O	S		X[5]
DSPOBJD				
DSPPFM	F O	S		X[5]
ENDCMTCTL	F	T		
FMTDTA	F	S T		X[11]
GRTOBJAUT		T		X
INZPFM	F O	S		X
	F	T[2]		X
MOVOBJ				
OPNDBF[6]	O	S		X
OPNQRYF	F	T	X	
OVRDBF	F	T	X	X
POSDBF	F	S		[7]
	F	T		X

Table 8. DDM-related CL commands (continued)

Command name	Related to file, object or both	Affects objects on source, target or both	OVRDBF command	SBMRMTCMD command[1]
QRYDTA				
RCVF		T		X
RCVNETF	F	T		
RGZPFM	F			X
RMVM	F	T		X
	F	T		X
RNMM				
RNMOBJ	F	T		X
ROLLBACK	F O	S T[8]		X
RSTLIB	F	S T		X[11]
RSTOBJ		S		X[9]
RTVDFUSRC	F O	S		X[9]
		T		X
RTVQRYSRC				
RVKOBJAUT		T		X
SAVCHGOBJ	F O	S		X
SAVLIB	O	S		X[9]
SAVOBJ		S		X[9]
	F O	S		X[9]
SBMDBJOB				
SNDNETF		T		X
STRBAS	F	T		X
STRBASPRC		T		X
STRCMTCTL	O	T		X
STRDBRDR	F	S T		X[11]
WRKJOB		T		X
WRKOBJLCK[10]	O			X[5]
	F O	S		X[5]

Notes:

1 The use of the SBMRMTCMD command is not valid with *any* of the commands in these charts unless the target server is an iSeries server or a System/38.

2 This member-related command can be used only if the target server is an iSeries server.

3 This member-related command can be used only if the target server is an iSeries server or a System/38.

4 These commands require intervention on the target server to load a tape and they might not produce the results expected.

5 When submitted to the target server, these commands produce output on the target server only; the output is not sent to the source server.

6 OPNDBF command: For more information about commitment control restrictions, see Commitment control support for DDM.

7 OVRDBF command: Although this command works when submitted on the SBMRMTCMD command to a target iSeries server or a System/38, it is *not* recommended.

8 RNMOBJ command: OBJTYPE*FILE must be specified.

9 When submitted to the target server, these commands require target server resources when tapes are used to produce the output.

10 WRKOBJLCK command: This command displays any locks on the DDM file, not the remote file.

11 This command will work, but its use is not recommended.

Related reference

"CL command considerations for DDM" on page 32
Both compiled control language (CL) programs and interactively entered CL commands can refer to DDM files.

"iSeries and System/38 target systems on the SBMRMTCMD command" on page 70
The SBMRMTCMD command can submit any CL command that can run in both the batch environment and using the QCAEXEC server program.

"Target iSeries-required file management commands" on page 101
These CL commands can be used only when the target server is another iSeries server or System/38.

"Member-related commands with DDM" on page 101
Database file operations that apply to a member can be used by DDM.

"Commitment control support for DDM" on page 27
iSeries applications can commit or roll back transactions on remote iSeries servers.

"Perform file management functions on remote servers" on page 119
i5/OS DDM supports creating, deleting, or renaming of files on a remote server.

Data description specifications considerations for DDM

Data description specifications (DDS), which is used to externally describe the fields and record formats, can also be used with DDM to describe the file and record formats of a remote file.

Related reference

"DDM architecture-related restrictions" on page 45
The items listed in this topic are DDM architecture-related restrictions. Therefore, application programs that use these items might have to be changed and recompiled before they can access remote files.

"DDM example 4: Access a file on System/36" on page 155
This topic shows how the pseudocode program for the previous task can be changed so a MASTER file on the System/36 in Dallas can be accessed in the same way as the MASTER files on the iSeries servers and System/38 in Example 3.

iSeries target considerations for DDM

This topic covers target considerations for DDM.

As with any database file, DDS might or might not be used to externally describe the attributes of the remote file when it is created on the remote iSeries server. If DDS is used, then the source server program uses those attributes when it accesses the remote file (by using the DDM file). If DDS is not used, then the file's attributes must be described in the program.

When a source server program that accesses a file on a target iSeries server is compiled (or recompiled), the existing DDM file is used to establish communications with the target server, and the remote file is actually accessed during compilation to extract its file and record attributes. Whether DDS is used to describe the file or not, level check identifiers are created during compilation and are included in the compiled program. These values are then used when the program is run and LVLCHK(*RMTFILE) is in effect for the DDM file.

Whether DDS is used to describe a remote iSeries file or not, a source server program can have its own field and record format definitions provided in the program, or the program can substitute the definitions of another source server file that is created using DDS. Either can be done if LVLCHK(*NO) is in effect in the DDM file or specified in an Override with Database File (OVRDBF) command used at program run time. LVLCHK(*NO) need only be used when the record format used on the source server is different than that of the remote iSeries file.

Non-iSeries target considerations for DDM

DDS can be used with a non-iSeries file only if the local iSeries program is compiled using a local iSeries file that has the same record format name as the DDM file being used.

After the program is compiled, the local file can be overridden by a DDM file that accesses the remote file. LVLCHK(*NO) must be specified in the DDM file or in an OVRDBF command.

If no DDS exists on the local server to describe the remote file, the program must describe the fields. The Display File Field Description (DSPFFD) command can be used to display the field attributes of the remote file. LVLCHK(*NO) should be specified in the DDM file or in an OVRDBF command.

If LVLCHK(*RMTFILE) is specified or assumed, the program must be compiled (or recompiled) using a DDM file that accesses the remote file. The iSeries server then creates a record format and fields for the remote file. The names of the fields that are created are of the type K*nnnnn* for keyed fields and F*nnnnn* for nonkeyed fields.

DDM-related DDS keywords and information

The information about DDS keywords that relates specifically to DDM is provided in this topic.

- Considerations for creating local files:
  - The following DDS keywords *cannot* specify the name of a DDM file: REFACCPTH, and FORMAT.
  - The DDS keywords REF and REFFLD *can* specify the names of DDM files to refer to remote files; however, the remote files must be on an iSeries server or a System/38. When a DDM file name is specified as the database file name in either keyword, it refers to the DDM file on the source server, and the referred to field name and record format name refer to a field and record format used in the remote file on the target server.
- Considerations for creating logical files when the remote server is not an iSeries server:
  - At least one key field must be specified in the record format for the logical file.
  - Only one file can be specified on the PFILE keyword.
  - SELECT and OMIT functions are not supported.
  - Logical join files are not supported.
  - Field names of remote physical files have the naming convention of F00001, F00002, F00003, and so forth (F*nnnnn*) for nonkeyed fields and K00001, K00002, K00003, and so forth (K*nnnnn*) for keyed fields.

    The exception to this naming convention is when the target server is a System/38 and the physical file was created locally. In this case, the field names are the same as the field names specified when the physical file was created.
  - All the fields defined for the logical file must be specified in the same order as defined in the physical file. This can be done by default.
  - The SST keyword can be used to access partial fields of the physical file. The use of two or more substring fields is required to define the entire physical field. In addition, the partial fields must be in the same order as defined in the substring field of the physical file.
  - The CONCAT keyword can be used to group physical file fields into one logical field. The concatenation order of the fields must be in the same order as defined in the physical file.
  - The fields of the physical file must be specified in the same order as defined in the physical file.
- Considerations for using the VARLEN DDS keyword when creating files on a non-iSeries target server:
  - Target server must support variable-length record files.
  - Only one variable-length field is allowed in the file format and it must be the last field.
  - The field with the VARLEN keyword must not be a key field.
- PFILE and JFILE considerations for creating remote files:
  - The record format name specified for the physical file in the DDM file on the JFILE or PFILE keyword must be the same name as the DDM file that represents the remote physical file.

– When creating a logical file, the file specified on PFILE or JFILE must be a DDM file, and the location for each physical file in the DDM file on the JFILE or PFILE keyword must be the same as the location of the DDM file for the logical file. In other words, the physical files and logical file must be on the same remote server.

If the remote server is a release 1.0 or 1.2 iSeries server, attempting to create a file using the FCFO keyword will fail.

- When the server is not an iSeries server, these keywords are either ignored or not supported for *logical* files:

ABSVAL	EDTCDE	LIFO
ACCPTH	EDTWRD	NOALTSEQ
ALIAS	FCFO	RANGE
ALL	FLTPCN	REFSHIFT
ALTSEQ	FORMAT	RENAME
CHECK	JDFTVAL	SIGNED
CMP	JDUPSEQ	TEXT
COLHDG	JFILE	TRNTBL
COMP	JFLD	VALUES
DIGIT	JOIN	ZONE
DYNSLT	JREF	

- When the server is not an iSeries server, these keywords are either ignored or not supported for *physical* files:

ABSVAL	EDTCDE	RANGE
ALTSEQ	EDTWRD	REFSHIFT
CHECK	FCFO	SIGNED
CMP	FLTPCN	TEXT
COLHDG	FORMAT	VALUES
COMP	LIFO	ZONE
DIGIT	NOALTSEQ	

Related reference

"CRTPF (Create Physical File) command" on page 90
The Create Physical File (CRTPF) command can be used to create files on the source and target servers through the SYSTEM parameter.

DDM user profile authority

iSeries server users are not allowed to perform functions equivalent to CL commands on remote iSeries servers using DDM without the proper command authorization.

The user profiles associated with the target jobs must have *OBJOPR authority to the following CL commands to start the equivalent operation on the remote iSeries server:

Command name	Descriptive name
ADDLFM	**Add Logical File Member**
ADDPFM	**Add Physical File Member**
ALCOBJ	**Allocate Object**
CHGLF	**Change Logical File**
CHGLFM	**Change Logical File Member**
CHGPF	**Change Physical File**
CHGPFM	**Change Physical File Member**

Command name	Descriptive name
CRTLF	Create Logical File
CRTPF	Create Physical File
DLTF	Delete File
INZPFM	Initialize Physical File Member
RGZPFM	Reorganize Physical File Member
RMVM	Remove Member
RNMM	Rename Member
RNMOBJ	Rename Object

Operating considerations for DDM

This topic provides task-oriented information and examples that describe various aspects of DDM operation considerations.

This topic tells how the iSeries server functions, both as a source or target server, when it communicates with another iSeries server to perform remote file processing. It also describes the significant differences when an iSeries server is communicating with another server that is not an iSeries server.

Note: Although this topic contains information about servers other than the iSeries server, it does not contain all the information that the other server types using DDM might need to communicate with an iSeries server. For complete information about how DDM is used with a particular remote server, refer to that server's documentation.

Related concepts

"Additional DDM concepts" on page 13
Most users of DDM will not need the information in the remainder of these topics; it is intended primarily for experienced programmers who need to know more about DDM.

Related reference

"CL command considerations for DDM" on page 32
Both compiled control language (CL) programs and interactively entered CL commands can refer to DDM files.

Access files with DDM

These topics describe the types of files supported by an iSeries server, when the DDM file and remote file must exist, and how to specify the names of remote files. Also included are examples and considerations for iSeries-to-iSeries and iSeries-to-System/36 file accessing.

Types of files supported by i5/OS DDM

i5/OS DDM supports all iSeries file types when the target server is another iSeries server.

If the target server is not an iSeries server, the corresponding file types might be known by different names on that server. The following table shows the iSeries equivalents of non-iSeries files and DDM architecture files:

iSeries types	Non-iSeries and DDM architecture types
Non-keyed physical file	Sequential (or direct) access file
Keyed physical file	Keyed access file
Logical file	Logical file

The following list describes the considerations that apply to the types of files supported by an iSeries server.

- iSeries multiple-format logical files are not supported by DDM when the source or target server is neither an iSeries server nor a System/38.
- For target physical (sequential or direct) files, if a record number is specified that is past the end of the file, the file is not extended and an error occurs.
- For target nondirect sequential files, the Clear Physical File Member (CLRPFM) command does not prepare a file member with deleted records.
- DDM files can be used as data files or source files by high-level language (HLL) programs. However, when a DDM file is used as a source file, the target server must be an iSeries server or a System/38 and the remote file associated with the DDM file must be defined on the target server as a source file. That is, the remote file must have been created on the target iSeries server or the target System/38 as FILETYPE(*SRC) by the Create Physical File (CRTPF) command or with FMTOPT(*CVTSRC) specified on the Copy File (CPYF) command.

  Related reference

  "Source file commands" on page 103
  If the target server is an iSeries server or a System/38, these CL commands can support a DDM file as a source file (on the SRCFILE parameter).

Existence of DDM file and remote file

A file on a target server cannot be accessed for any type of operation (such as open, read, write, or display) unless a DDM file associated with the remote file already exists on the source server.

However, the remote file does not need to exist at the time that the DDM file is created or changed using the Create DDM File (CRTDDMF) command or the Change DDM File (CHGDDMF) command, because the remote file is not referred to until the DDM file is actually opened for access.

Rules for specifying target server file names for DDM

The rules for specifying the name of a DDM file (on the local iSeries server) are the same as for any other file type on an iSeries server. The rules, however, for specifying the name of a remote file depend on the type of target server.

A remote file name can be specified only on the RMTFILE parameter of the Create DDM File (CRTDDMF) and Change DDM File (CHGDDMF) commands. The following list is the maximum number of characters that can be used on the RMTFILE parameter to specify a remote file name:

- For the iSeries server (database management): 33 characters. This maximum can occur when a full name is specified that includes a library qualifier and a member name. For example:

  ```
  LIBRARY123/FILE123456(MEMBER1234)
  ```

  The value DM can be added to the name to specify that this is a data management file. There can be one or more blanks between the name and DM. This is the default.
- For the iSeries server (folder management services): 76 characters. This maximum can occur when a fully qualified path name (consisting of 76 characters) is specified. For example:

  ```
  /Path123/Path223/Path323/Path423/
     Path523/Path623/Path723/Path823/Path923/DOC1 FMS
  ```

  The value FMS specifies that this is a folder management object. There can be one or more blanks between the name and FMS.
- For System/38: 33 characters. This maximum can occur when a full name is specified that includes a library qualifier and a member name. For example:

  ```
  FILE123456.LIBRARY123(MEMBER1234)
  ```
- For System/36 and CICS: 8 characters. For example:

  ```
  FILE1234
  ```

- For other systems: 255 characters is the maximum length allowed by the DDM architecture. The actual maximum length and syntax are determined by the target server.

Related reference

"iSeries server and System/36 DDM differences" on page 206
This topic consists of a list of differences between the iSeries server and System/36.

"iSeries server and System/38 DDM differences" on page 207
This topic consists of a list of differences between the iSeries server and System/38.

Target iSeries file names for DDM:

As with local files, every iSeries remote file, library name, or member must begin with an alphabetic character (A through Z, $, #, or @) and can be followed by no more than 9 alphanumeric characters: A through Z, 0 through 9, $, #, @, _, or period (.). No name can exceed 10 characters. Blanks are not allowed in iSeries names.

The use of an extended name allows additional graphic characters to be included in quotation marks (''). The extended name also cannot exceed 10 characters, but quotation marks are included with the name, thereby limiting the number of graphic characters to 8. Lowercase letters remain lowercase letters. Examples of extended names are as follows:

```
"Test.Job"
"()/+="
```

When an iSeries server is the target server, the file name can be specified in various forms, as shown in the following examples.

library-name
Specifies the name of the library that contains the remote file. *LIBL causes the library list of the job on the target server to be searched for the specified file name. *CURLIB specifies the current library on the remote server.

remote-file-name
Specifies the name of a database file (physical, logical, or source file) on the target iSeries server.

***NONSTD**
Specifies, for an iSeries target, that a member name is being included with the name of the remote file. The value *NONSTD *must* precede the full name, and the full name must be enclosed in single quotation marks and be in all uppercase.

Note: If you press F4 (Prompt) when on the Create DDM File or Change DDM File displays, and specify the *NONSTD value with the remote file name abcde, the server converts abcde to 'ABCDE' (all uppercase) and the request is processed. However, if there is a slash or parenthesis in the remote file name, the system puts single quotation marks around the name but does not convert it to uppercase.

Therefore, if you are using the *NONSTD value for the remote file name and the target server requires uppercase file names, type the remote file name in uppercase characters even when using F4 (Prompt).

member-name
Specifies the name of the member in the remote file. The member name must be enclosed in parentheses and immediately follow the file name (with no space). If no member name is specified, then *FIRST is assumed and the first (or only) member in the file is accessed. This is the oldest (or only) member in the file.

*LAST is supported only on the Override with Database File (OVRDBF), Clear Physical File Member (CLRPFM), Initialize Physical File Member (INZPFM), Reorganize Physical File Member (RGZPFM), Open Database File (OPNDBF), and Open Query File (OPNQRYF) commands. *LAST is the newest (or only) member in the file.

The examples here show valid iSeries remote file names:

```
CUSTMAST
PRODLIB/CUSTMAST
*NONSTD 'CUSTMAST(MBR1)'
*NONSTD '*LIBL/CUSTMAST(MBR2)'
*NONSTD 'PRODLIB/CUSTMAST(MBR3) DM'
*NONSTD 'PRODLIB/CUSTMAST(*FIRST)'
```

Target non-iSeries file names for DDM:

For non-iSeries remote file names, the name must be in the form required by the target server.

If special characters are used in the remote file name, *NONSTD and single quotation marks must be used to specify the name, as shown in how to specify an iSeries member name. If the name string contains no more than 10 characters and no special characters, it can be entered without the *NONSTD value and the single quotation marks.

Use location-specific file names for commonly named files for DDM:

When multiple servers are involved in a network, naming DDM files with location-specific file names can reduce confusion about which target server is being accessed for a file that has a commonly used name.

For example, for an inventory file that may be named INVEN on multiple servers, using location-specific names such as NYCINVEN, STLINVEN, and DALINVEN for the DDM files on the local server to access files in New York City, St. Louis, and Dallas helps you to access the correct file.

Using an abbreviation or code that identifies the destination target server as part of the DDM file names makes it easier to remember where the desired remote file is located.

For non-iSeries remote files that have record formats, using the same name for the DDM file as for the record format can also be useful.

Examples: Access iSeries DDM remote files (iSeries-to-iSeries)

The examples show how access to a DDM file becomes an indirect reference (by using DDM) to the actual file on some other server. These examples are iSeries-to-iSeries examples.

Note: All of these examples assume that the DDM file on the local iSeries server is named DDMLIB/RMTCAR and that it is associated with a remote file named SALES/CAR on an iSeries server in Chicago.

Create a DDM file to access a remote file

```
CRTDDMF   FILE(DDMLIB/RMTCAR)  RMTFILE(SALES/CAR)
   RMTLOCNAME(CHICAGO)  TEXT('Chicago file SALES/CAR')
```

This command creates a DDM file named RMTCAR and stores it in the DDMLIB library on the local server. The remote file to be accessed is the CAR database file in the SALES library on the Chicago server. (The remote file is *not* accessed at the time the Create DDM File (CRTDDMF) command is used to create the DDM file. The existence of the file SALES/CAR is not checked when the DDM file is created.) Later, when the DDM file is accessed by a local program, the remote location CHICAGO is used by DDM to access the SALES/CAR file on the Chicago server.

Copy a local file to a remote file

```
CPYF   FROMFILE(QGPL/AUTO)  TOFILE(DDMLIB/RMTCAR)
```

This command copies the data from the AUTO file in the QGPL library on the local server into a remote file named SALES/CAR on the Chicago server, by using the DDM file DDMLIB/RMTCAR.

Allocate a remote file and member for use

```
ALCOBJ   OBJ((DDMLIB/RMTCAR  *FILE  *EXCL))
```

The Allocate Object (ALCOBJ) command is used to allocate (lock) both the DDM file (RMTCAR) on the source server and the first member of the remote file (as well as the file itself) on the target server. In effect, the command

```
ALCOBJ   OBJ((SALES/CAR  *FILE  *EXCL  *FIRST))
```

is run on the target server.

Override a local file with a DDM file

```
OVRDBF  FILE(FILEA)  TOFILE(DDMLIB/RMTCAR)
   POSITION(*RRN 3000)
```

This command overrides the database file FILEA with the DDM file RMTCAR, stored in the DDMLIB library. Both files are on the source server. Whatever remote file is identified in the DDM file (in this case, SALES/CAR on the Chicago system) is the file actually used by the source server program. When the remote file is opened, the first record to be accessed is record 3000.

Display records in a remote file

```
DSPPFM   FILE(DDMLIB/RMTCAR)
```

This command displays the records in the first member of the remote file SALES/CAR, which is associated with the DDM file DDMLIB/RMTCAR.

Display the object description of a DDM file

```
DSPOBJD   OBJ(DDMLIB/RMTCAR)  OBJTYPE(*FILE)
```

This command displays, on the local server, the object description of the RMTCAR DDM file. No reference is made by this command to the associated remote file on the Chicago server.

Display the file description of a DDM file

```
DSPFD  FILE(DDMLIB/RMTCAR) TYPE(*ATR) FILEATR(*DDM)
   SYSTEM(*LCL)
```

This command displays, on the source server, the file description of the DDM file named RMTCAR in the DDMLIB library. As indicated by the TYPE parameter, the attributes of the DDM file are displayed. Only the DDM file's attributes are displayed because FILEATR(*DDM) is specified.

Because SYSTEM(*LCL) is specified, the attributes of the DDM file are displayed and the remote server is not accessed. If SYSTEM(*RMT) is specified, the attributes of the associated remote file are displayed. If *RMT or *ALL is specified, the remote server is accessed to get the attributes of the remote file.

Delete a DDM file

```
DLTF  FILE(DDMLIB/RMTCAR)  SYSTEM(*LCL)
```

This command deletes the DDM file on the local server. Again, no reference is made to the associated SALES/CAR file on the Chicago server. If SYSTEM(*RMT) or SYSTEM(*FILETYPE) is specified, SALES/CAR on the Chicago server would be deleted.

Example: Access System/36 DDM remote files (iSeries-to-System/36)

Of the command examples given in the previous topic (showing iSeries-to-iSeries examples), all except the first example can be coded the same way for accessing a file on a System/36.

That is, if the remote file name SALES/CAR is changed to CAR to meet the System/36 naming conventions, all the commands (except the first) can be used without change to access a remote System/36 file instead of an iSeries file.

The first example from the topic Examples: Access iSeries DDM remote files (iSeries to iSeries) is recorded here to access a remote System/36 file. Besides changing the remote file name, another parameter that should be coded is LVLCHK(*NO).

```
CRTDDMF   FILE(DDMLIB/RMTCAR)  RMTFILE(*NONSTD 'CAR')
   RMTLOCNAME(CHICAGO)  TEXT('Chicago file CAR on S/36')
   LVLCHK(*NO)
```

This command creates a DDM reference file named RMTCAR and stores it in the DDMLIB library on the local iSeries server. The remote file to be accessed is the CAR file on the System/36 named CHICAGO. LVLCHK(*NO) is specified to prevent level checking because the level identifiers created for the System/36 file do not match those in the program when it accesses the file.

Access members with DDM

Members are supported for database I/O operations only if the target server is an iSeries server or a System/38. Members are not supported if the target server is neither an iSeries server nor a System/38.

Members can be locked before use, using the Allocate Object (ALCOBJ) command if the target server is an iSeries server or a System/38.

The DDM file itself does not have members like a database file. However, if a member is identified on the source server (for example, using the Override with Database File (OVRDBF) command) and the target server is an iSeries server or a System/38, that member name is used to identify a member in the target server's file. When the target server is neither an iSeries server nor a System/38, and if the member name is specified as *FIRST, or in some cases *LAST, or the file name is the same as the member name, then the RMTFILE parameter values in the DDM file are sent without change. This allows file access on servers that do not support members.

If the member name is other than *FIRST or in some cases *LAST, or the file name is different from the member name (for example, when the file is opened) and the target server does not support members, an error message is sent to the requesting program and the function is not performed.

Example: Access DDM remote members (iSeries server only)

These examples show how access to a DDM file becomes an indirect reference (by using DDM) to a member of a file on a remote iSeries server. These examples are iSeries server-to-iSeries server examples.

```
CRTDDMF  FILE(DDMLIB/RMTCAR)  RMTFILE(SALES/CAR)
   RMTLOCNAME(CHICAGO)
OVRDBF   FILE(FILE1)  TOFILE(DDMLIB/RMTCAR)  MBR(TEST1)
OVRDBF   FILE(FILE2)  TOFILE(DDMLIB/RMTCAR)
```

This example shows the creation of the same DDM file as in the previous examples. Then, OVRDBF commands are used to override two local files named FILE1 and FILE2 with the local DDM file RMTCAR. When an application program attempts to open the files, the DDM file DDMLIB/RMTCAR is opened twice instead. (FILE1 and FILE2 are not opened.)

After communications are established with the correct target server, the target server's TDDM opens the remote file SALES/CAR twice (two recursions) and opens two different (in this case) members in that file: member TEST1 and member *FIRST (the first member). This example requires only one DDM conversation and one target job because both open operations use the same DDM file and, therefore, the same location.

```
CLRPFM   FILE(DDMLIB/RMTCAR)  MBR(FRED)
```

This command clears, by using the DDM file named DDMLIB/RMTCAR, member FRED of the file SALES/CAR on the target server.

Related reference

"OVRDBF (Override with Database File) command" on page 94
The Override with Database File (OVRDBF) command can be used with DDM to override (replace) a local database file named in the program with a DDM file; the DDM file causes the associated remote file to be used by the program instead of the local database file.

Example: DDM file that opens a specific member

A specific file member can be specified in the RMTFILE parameter, which is used only on the Create DDM File (CRTDDMF) and Change DDM File (CHGDDMF) commands, by using the *NONSTD value followed by the file, library, and member name.

This allows an application program to process a member other than the first member (*FIRST) without using file overrides. However, if the program requires redirection to more than one member, overrides should be used. Also, programs that already use overrides to specify members of local files should continue to do so, even if overrides to remote files are also used; otherwise, programs that worked locally would no longer do so. If the RMTFILE parameter contains a member name and an override with a different member name is in effect, the file open requests fails.

Note: If you press F4 (Prompt) on the Create DDM File or Change DDM File display, and specify the *NONSTD value with the remote file name abcde, the server converts abcde to 'ABCDE' (all uppercase) and the request is processed. However, if there is a slash or parenthesis in the remote file name, the server puts single quotation marks around the name but does not convert it to uppercase. Therefore, if you are using the *NONSTD value for the remote file name and the target server requires uppercase file names, type the remote file name in uppercase characters even when using F4 (Prompt).

```
CRTDDMF   FILE(DDMLIB/RMTCAR)  RMTFILE(*NONSTD
   'SALES/CAR(JULY)')  RMTLOCNAME(CHICAGO)
```

When a program opens the DDM file named RMTCAR on the source server DDMLIB library, the target iSeries server opens the member JULY in the file SALES/CAR.

Work with access methods for DDM

Access methods control what subsets of functions can be performed after a particular remote file is opened. This might mean that an iSeries program, or a group of programs sharing a non-iSeries file, cannot do all the same operations that are possible using a file that is on the local iSeries server.

For example, assume that an iSeries application program opens a keyed file with SHARE(*YES) and performs keyed I/O operations. It then calls another program that does relative record number operations using the same open data path (ODP) (because SHARE was specified). *Relative record numbers* specify the relationship between the location of a record and the beginning of a database file, member, or subfile. If the first program is redirected by an Override with Database File (OVRDBF) command to use a remote keyed file on a System/36, this scheme no longer works. If a *keyed* access method is selected, record number operations fail. If a *record number* access method is selected, keyed operations fail.

Notice that when both source and target servers are iSeries servers, access methods are not used. A potential problem exists when the target server is neither an iSeries server nor a System/38. Notice also that the combined-access access method (*COMBINED) is not supported by System/36, and probably not by any target other than an iSeries server or System/38.

Access intents

When a program opens a file, it must specify how it intends to work with the records in the file: read, add, update, delete, or a combination of these.

Of course, to successfully perform these operations, the job, user or both running the program must have the corresponding data authorities. The iSeries server does not check to make sure all data authorities exist when the file is opened, but it does check for each required data authority when the corresponding I/O operation is done using the file. The System/36 does check these data authorities at open time; therefore, a program may no longer work using a remote file on a System/36, even though the requester's data authorities to the remote file are the same as for a local file (which will work).

For example, assume that a program is used by two groups of users on an iSeries server to access the same local iSeries file. Group A has only *READ authority, while group B has *READ, *ADD, and *UPDATE. The program always opens the file for *READ, *ADD, and *UPDATE. But it has a *read only* logic path that is used when a member of group A calls the program. In this way, no authority exceptions are encountered, even though exceptions would be created if members of group A attempted to add or update records. Now, if this program is redirected to a remote System/36 file to which members of both user groups have the same data authorities as they had to the local iSeries file, the program might not work for members of group A. This is because the System/36 may reject requests to open the file when the requester does not have data authorities matching those specified in the access intent list accompanying the open request.

Key field updates

An iSeries program is allowed to change any part of a data record including key fields.

The exception to this is an ILE COBOL program because the ILE COBOL language does not allow key field changes. A System/36 program cannot change primary key fields in a record, regardless of the access method specified when the file is opened. Logical file key fields can be changed under some circumstances, but primary key fields can never be changed.

This means that an ILE RPG program, for example, that routinely changes key fields in a local keyed file might fail when it is redirected to a remote keyed file on a System/36 (or other system with similar restrictions). Several different errors might be returned by the DDM target, depending on the access method or access path being used when the key field change is attempted.

Deleted records

On the iSeries server, a record is marked as deleted by the server.

This is done either when an active record is deleted by an application or when a file is created with deleted records (for example, with the Initialize Physical File Member (INZPFM) command). A record that is added to a file or changed in a file is never marked as deleted, unless a subsequent delete operation is performed. On some other servers, like the System/36, a special data value in the record might be used to indicate deleted status. For example, if a record contains all hex FFs, it might be considered deleted.

This means that an iSeries application normally used to add or change records in a local file might encounter errors when attempting these operations with a remote file on a server that is neither an iSeries server nor a System/38. If the application happens to supply a record that is considered deleted by the target DDM server, the target might reject the add-or-change request.

Blocked record processing

If SEQONLY is used to block records sent to a remote server, the records are not sent until the block is full. If a source job is canceled before a block is sent, the records in the block are lost. If blocking is used, the user should make sure a force end of data or close of the file is done before canceling the source job.

Variable-length records

If your iSeries source server is running OS/400 Version 2 Release 1 Modification 1, DDM supports variable-length record files as defined in the DDM architecture.

You can use DDM on your iSeries server to open variable-length record files on target systems that are not iSeries or S/38 servers. (Initially you can open variable-length record files if you are not opening the file for updating.) For subsequent read operations, variable-length records are padded with blanks to the maximum record length of the file. Trailing blanks are removed on write operations.

If your iSeries source server is running OS/400 Version 2 Release 2 in addition to the Version 2 Release 1 Modification 1 support mentioned earlier, iSeries variable-length record access is supported using DDM. Variable-length records can be used when opening a variable-length record file on target servers that are not iSeries or System/38 servers. For subsequent read operations against files opened with variable-length records, variable-length records are padded with blanks to the maximum record length of the file. Also, the actual record length (maximum record length of file minus the number of padded blanks) is appended to the end of each record. For write operations, the actual record length is used to determine the length of the variable-length record to send to the target server. No counting of trailing blanks is necessary to determine the actual length of record data.

The target DDM iSeries servers running at Version 2 Release 2 also support variable-length record files. A variable-length record file can be created on the iSeries target server as a result of a create file request.

Related reference

"DDM commands and parameters" on page 165
This topic classifies DDM commands and parameters.

Other DDM-related functions involving remote files

Besides accessing remote files for data record I/O operations, other operations related to remote files can be performed. These are briefly described in these topics.

Perform file management functions on remote servers

i5/OS DDM supports creating, deleting, or renaming of files on a remote server.

The Submit Remote Command (SBMRMTCMD) command can be used to submit these types of file management commands, or other CL commands, to the target server so they can run on that server. The Submit Network Job (SBMNETJOB) command or display station pass-through can also be used, without the need for DDM.

Note: The CL commands in "Target iSeries-required file management commands" on page 101, "Member-related commands with DDM" on page 101, and "Source file commands" on page 103 do not need to be used with the SBMRMTCMD command; they can run directly on the target server by specifying a DDM file name on the CL command itself.

Related reference

"SBMRMTCMD (Submit Remote Command) command" on page 69
The Submit Remote Command (SBMRMTCMD) command submits a command using DDM to run on the target server.

"Examples: Code DDM-related tasks" on page 146
The examples in this topic collection are based on representative application programs that might be used for processing data both on the local iSeries server and on one or more remote servers.

"Object-oriented commands with DDM" on page 99
The DDM file object on the source iSeries server can be accessed by these object-oriented CL commands. These commands work with DDM files as they normally do with any other files on the local server.

"DDM-related CL command summary charts" on page 104
This topic shows summary charts containing most of the control language (CL) commands used with DDM.

Lock files and members for DDM

Putting object locks on DDM files and their associated remote files requires special consideration.

Allocate Object (ALCOBJ) and Deallocate Object (DLCOBJ) commands:

The ALCOBJ command locks DDM files on the source server and the associated remote files on the target servers.

When the target is an iSeries server or a System/38, resulting locks on the remote files are the same as if the files were local files. When the target is neither an iSeries server nor a System/38, equivalent locks are obtained, although the target server may promote the lock to a stronger lock condition than was specified on the ALCOBJ command.

Note: On servers that are neither iSeries nor System/38 target servers, remote *files* are locked with the specified lock condition, and on iSeries and System/38 target servers only, remote *members* are locked with a minimum specified lock condition. (iSeries or System/38 remote *files* are locked with shared-read locks.)

Related reference

"ALCOBJ (Allocate Object) command" on page 82
When the name of a DDM file is specified on the Allocate Object (ALCOBJ) command on the source server, the command allocates the DDM file on the source server and its associated file or file member on a target server.

"DLCOBJ (Deallocate Object) command" on page 92
When the name of a DDM file is specified on the Deallocate Object (DLCOBJ) command on the source server, the command deallocates the DDM file on the source server and its associated file or file member on a target server.

Work with Job (WRKJOB) and Work with Object Locks (WRKOBJLCK) commands:

For both the WRKOBJLCK command and menu option 12 (Work with locks, if active) of the WRKJOB command, only the locks held for the local DDM files are shown, *not* locks held for the remote files (or for their members).

If locked, DDM files are always locked as shared read (*SHRRD), regardless of the lock conditions used for their associated remote files or members.

Related reference

"WRKJOB (Work with Job) command" on page 96
The Work with Job (WRKJOB) command can be used to display two DDM-related items.

"WRKOBJLCK (Work with Object Lock) command" on page 96
The Work with Object Lock (WRKOBJLCK) command can be used to display the object lock requests (held locks and pending locks) for DDM files. This command displays only the locks held for the local DDM files, not locks held for the associated remote files.

Control DDM conversations

Normally, the DDM conversations associated with a source server job are kept active until one of the conditions described in this topic is met.

1. All the DDM files and remote files used in the conversation are closed and unlocked (deallocated).
2. No other DDM-related functions like the use of the Submit Remote Command (SBMRMTCMD) command or the Display File Description (DSPFD) command (to display remote file information) are being performed.
3. No DDM-related function has been interrupted (by a break program, for example) while running.
4. The ENDCMTCTL command was issued (if commitment control was used with a DDM file).
5. No distributed relational database architecture-related functions are being performed.

6. The activation group, in which the DDM conversation was started, ends. A DDM conversation is not dropped when the activation group ends under the following conditions:
   - The DDM conversation is scoped to the job level.
   - The commitment control of the activation group is scoped to the job level, and a unit of work is outstanding. The conversation remains until the next job level commit or rollback, or until the job ends.
7. The job or routing step ends.

If *1*, *2*, and *3* are true and the source job or activation group has not ended, the conversation is considered to be *unused*, that is, the conversation is kept active but no requests are being processed.

DDM conversations can be active and unused because the default value of the DDMCNV job attribute is *KEEP. This is desirable for the usual situation of a source server program accessing a remote file for multiple I/O operations; these operations are handled one at a time, as shown in Figure 7 on page 16 and explained in the text following it.

If multiple DDM requests are to be made in a job and the DDM files are being continually opened and closed in the job, *KEEP should be used to keep an unused DDM conversation active. (However, as long as one DDM file remains open or locked, *KEEP has no effect.)

For source jobs that access remote files but do not access data records in them, it might be desirable, depending on the frequency of the file accesses, to automatically drop each DDM conversation at the completion of each file-related source job request. Whether the conversation in the source job is kept active or automatically dropped during the time a conversation is unused is determined by the DDMCNV job attribute value (*KEEP or *DROP).

Regardless of the value of the DDMCNV job attribute, conversations are dropped when one of the following things occurs:

- The job ends.
- The activation group ends. A DDM conversation is not dropped when the activation group ends under the following conditions:
  - The DDM conversation is scoped to the job level.
  - The commitment control of the activation group is scoped to the job level, and a unit of work is outstanding. The conversation remains until the next job level commit or rollback, or until the job ends.
- The job initiates a Reroute Job (RRTJOB) command.

Unused conversations within an active job can also be dropped by the Reclaim DDM Conversations (RCLDDMCNV) or Reclaim Resources (RCLRSC) command. Errors, such as communications line failures, can also cause conversations to drop.

Related concepts

"iSeries server as the source server for DDM" on page 13
When an application program or user in a source server job first refers to a DDM file, several actions occur as part of processing the request on the source server.

Related reference

"RCLDDMCNV (Reclaim DDM Conversations) command" on page 68
The Reclaim DDM Conversations (RCLDDMCNV) command is used to reclaim all DDM source server conversations that are not currently being used by a source job.

"RCLRSC (Reclaim Resources) command" on page 95
The Reclaim Resources (RCLRSC) command, like the Reclaim DDM Conversations (RCLDDMCNV) command, can be used to reclaim all DDM conversations that currently have no users in the job.

"DDMCNV parameter considerations" on page 97
The DDMCNV parameter is a job-related parameter that controls whether Advanced Program-to-Program Communication (APPC) or iSeries conversations in the job that is allocated for DDM use (that is, DDM conversations) are to be automatically dropped or kept active for the source job.

Display DDMCNV values (WRKJOB command):

To display the current value (*KEEP or *DROP) of the DDMCNV job attribute for your source job, you can use menu option 2 (Work with definition attributes) on the Work with Job (WRKJOB) Command display. You can also find out the value within a CL program by using the Retrieve Job Attributes (RTVJOBA) command.

Change DDMCNV values (CHGJOB) command:

To control whether the server is to automatically reclaim (or drop) DDM conversations in a source job whenever they become unused, the server default *KEEP can be changed to *DROP by using a Change Job (CHGJOB) command. If the value is left as *KEEP, the Reclaim DDM Conversations (RCLDDMCNV) or Reclaim Resources (RCLRSC) command can be used at any time to drop all DDM conversations (within that job only) that currently do not have any active users.

Reclaim DDM resources (RCLRSC and RCLDDMCNV commands):

When an iSeries user wants to ensure that the resources for all APPC conversations (including DDM conversations) that are no longer active are returned to the server, the Reclaim Resources (RCLRSC) command can be used.

To reclaim currently unused DDM conversations in a job, use the Reclaim DDM Conversations (RCLDDMCNV) command.

Related concepts

"Use CL and DDS with DDM" on page 66
This topic contains DDM-related information about specific iSeries control language (CL) commands, data description specifications (DDS) considerations, DDS keywords, and DDM user profile authority.

Control language

Display DDM remote file information

The CL commands Display File Description (DSPFD) and Display File Field Description (DSPFFD) can be used by an iSeries source server user to display the attributes of one or more DDM files on the source server, or to display the attributes of one or more remote files on a target server.

Related reference

"DSPFD (Display File Description) command" on page 93
The Display File Description (DSPFD) command can be used to display (on the source server) the attributes of the DDM file on the source server, the remote file on the target server, or both the DDM file and the remote file. As with local files, the attributes of multiple DDM files, multiple remote files, or both can be displayed by the same command.

"DSPFFD (Display File Field Description) command" on page 93
The Display File Field Description (DSPFFD) command can be used to display the file, record format, and field attributes of a remote file. To display the remote file attributes, however, you must enter the name of the DDM file associated with the remote file, not the name of the remote file.

Display DDM remote file records

The Display Physical File Member (DSPPFM) command can be used to display a remote file on a target server.

For performance reasons, however, whenever possible, you should use display station pass-through to sign on the remote server, and display the file directly. When display station pass-through is used, only the display images are transmitted over the communications line. When DDM is used to access the remote file, each record is transmitted separately over the line, which requires many more transmissions.

If pass-through cannot be used (for example, if the remote file is not on an iSeries server, a System/38, or a System/36, or if pass-through is not configured on your server), direct record positioning rather than relative positioning should be used whenever possible. For example, if record number 100 is being displayed and you want to see record number 200 next, that record is accessed faster if you enter 200 in the control field instead of +100. The results are the same, unless the file contains deleted records.

Related concepts

"Performance considerations for DDM" on page 134
These topics provide information to help you improve performance when using DDM and also provide some information about when to use something other than DDM to accomplish some functions.

Coded character set identifier with DDM

Support for the national language of any country requires the proper handling of a set of characters.

A cross-system support for the management of character information is provided with the Character Data Representation Architecture (CDRA). CDRA defines the coded character set identifier (CCSID) values to identify the code points used to represent characters, and to convert these codes (character data), as needed to preserve their meanings.

Keep these considerations in mind when you are using CCSIDs with DDM:

- Data is converted to the process CCSID of the source job if both the source and target servers support CCSIDs.
- Data is not converted if one server is an iSeries server that supports CCSIDs and the other server is any other server that does not support CCSIDs.
- A file created on an iSeries target server by any source server that does not support CCSIDs is always created with CCSID 65535.
- The SBMRMTCMD (Submit Remote Command) command can be used to change the file CCSID on an iSeries target server by specifying the CHGPF (Change Physical File) command and the CCSID parameter.

Use of object distribution

Although DDM file names can be specified on the Send Network File (SNDNETF) and Receive Network File (RCVNETF) commands, these commands should be run, whenever possible, on the server where the data actually exists. Therefore, if both servers are iSeries servers and both are part of a SNADS network, *object distribution* can be used instead of DDM to transfer the data between them.

- The SNDNETF command should run directly on the server that contains the data being sent. If necessary, the Submit Remote Command (SBMRMTCMD) or Submit Network Job (SBMNETJOB) command can be used to submit the SNDNETF command to the server where the data exists.

  Note: Another way to use the SNDNETF command without using DDM is to run it on the target server using display station pass-through.
- The RCVNETF command must be run on the server where the data has been sent. If necessary, a DDM file can be referred to on the RCVNETF command to place the data on another server. However, if possible, you should arrange to have the data sent to the server where the data is to be used, to avoid using a DDM file.

For both sending and receiving operations, the file types of the data files must match and can only be a save file or a physical database file. If DDM is being used, however, the file being transferred cannot be a save file.

Related reference

"Batch file processing with DDM" on page 138
Consider these items when using batch file processing with DDM.

Use of object distribution with DDM

You can also use *both* SNADS (on iSeries servers) and DDM (on iSeries servers and non-iSeries servers) to transfer files between iSeries servers and servers that are not part of a SNADS network but have DDM installed.

Although a System/36 might have SNADS, it cannot be used for iSeries object distribution.

For example, if an i5/OS DDM file refers to a file on a System/36, the iSeries server can use the SNDNETF command to send the file to another iSeries server using object distribution. Similarly, if a file has been sent to an iSeries server, the RCVNETF command can be used to receive the file onto a System/36 using DDM.

See the *SNA Distribution Services* manual on the V5R1 Supplemental Manuals Web site.

Manage the TCP/IP server

These topics describe how to manage the DRDA and DDM server jobs that communicate using sockets over TCP. It describes the subsystem in which the server runs, the objects that affect the server, and how to manage those resources.

The DRDA and DDM TCP/IP server that is shipped with the i5/OS program does not typically require any changes to your existing system configuration in order to work correctly. It is set up and configured when you install i5/OS. At some time, you might want to change the way the system manages the server jobs to better meet your needs, solve a problem, improve the system's performance, or look at the jobs on the system. To make such changes and meet your processing requirements, you need to know which objects affect which pieces of the system and how to change those objects.

These topics describe, at a high level, some of the work management concepts that need to be understood in order to work with the server jobs and how the concepts and objects relate to the server.

Related concepts

Managing work

Related reference

"Create a DDM file using TCP/IP" on page 9
You can create a DDM file that uses TCP/IP as the communication protocol for connecting with the remote server.

DDM terminology

The same server is used for both DDM and DRDA TCP/IP access to DB2 Universal Database for iSeries.

For brevity, the term *DDM server* will be used rather than *DRDA/DDM server* in the following discussion. Sometimes, however, it may be referred to as the *TCP/IP server*, the *DRDA server*, or the *server* when the context makes the use of a qualifier unnecessary.

The DDM server consists of two or more jobs, one of which is what is called the DDM *listener* (or daemon), because it listens for connection requests and dispatches work to the other jobs. The other job or jobs, as initially configured, are prestart jobs which service requests from the DRDA or DDM client after the initial connection is made. The set of all associated jobs, the listener and the server jobs, are collectively referred to as the *DDM server*.

The term *client* is used interchangeably with *DRDA Application Requester* (or AR) in the DRDA application environment. The term client will be used interchangeably with *DDM source system* in the DDM (distributed file management) application environment.

The term *server* is used interchangeably with *DRDA Application Server* (or AS) in the DRDA application environment. The term server will be used interchangeably with *DDM target system* in the DDM (distributed file management) application environment.

TCP/IP communication support concepts for DDM

Several concepts pertain specifically to the TCP/IP communications support used by DRDA and DDM.

Establish a DRDA or DDM connection over TCP/IP:

To initiate a DDM server job that uses TCP/IP communications support, the DRDA application requester or DDM source system will connect to the well-known port number 446 or 447.

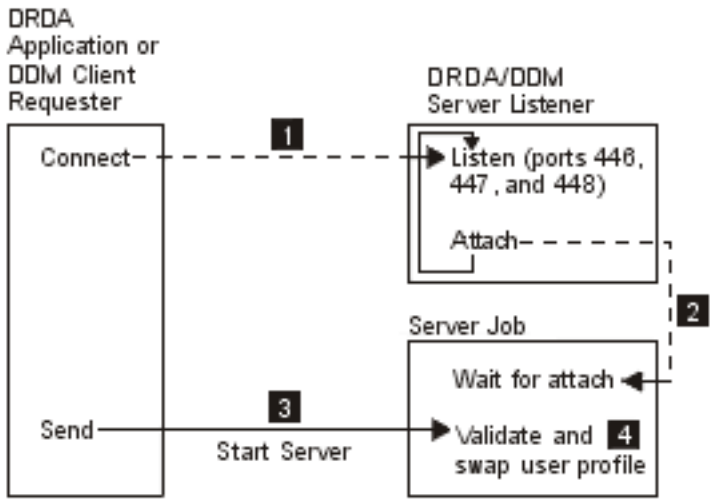


Figure 15. DRDA/DDM TCP/IP server

The DDM server also listens on port 448, but only for use with Secure Sockets Layer (SSL) connections, which are not supported by DB2 Universal Database for iSeries application requesters or DDM clients. 1. The DDM listener program must have been started (by using the STRTCPSVR SERVER(*DDM) command, for example) to listen for and accept the client's connection request. The DDM listener, on accepting this connection request, will issue an internal request to attach the client's connection to a DDM server job 2. This server job might be a prestarted job or, if the user has removed the QRWTSRVR prestart job entry from the QUSRSYS or user-defined subsystem (in which case prestart jobs are not used), a batch job that is submitted when the client connection request is processed. The server job will handle any further communications with the client.

The initial data exchange that occurs includes a request that identifies the user profile under which the server job is to run 3. After the user profile and password (if it is sent with the user profile ID) have been validated, the server job will swap to this user profile as well as change the job to use the attributes, such as CCSID, defined for the user profile 4.

The functions of connecting to the listener program, attaching the client connection to a server job and exchanging data and validating the user profile and password are comparable to those performed when an APPC program start request is processed.

DDM listener program:

The DDM listener allows client applications to establish TCP/IP connections with an associated server job by handling and routing inbound connection requests. After the client has established communications with the server job, there is no further association between the client and the listener for the duration of that connection.

The DDM listener program runs in a batch job. There is a one-to-many relationship between it and the actual server jobs; there is one listener and potentially many DDM server jobs. The server jobs are normally prestart jobs. The listener job runs in the QSYSWRK subsystem.

The DDM listener must be active in order for DRDA application requesters and DDM source systems to establish connections with the DDM TCP/IP server. You can request that the DRDA listener be started automatically by either using the CHGDDMTCPA AUTOSTART(*YES) CL command or through iSeries Navigator. In iSeries Navigator, navigate to the DDM settings: **Network → Servers → TCP/IP**. This will cause the listener to be started when TCP/IP is started. When starting the DRDA listener, both the QSYSWRK subsystem and TCP/IP must be active.

Start TCP/IP Server (STRTCPSVR) CL command:

The Start TCP/IP Server (STRTCPSVR) command, with a SERVER parameter value of *DDM or *ALL, is used to start the listener.

End TCP/IP Server (ENDTCPSVR) CL command:

The End TCP/IP Server (ENDTCPSVR) command ends the DDM server.

If the DDM listener is ended, and there are associated server jobs that have active connections to client applications, the server jobs will remain active until communication with the client application is ended. Subsequent connection requests from the client application will fail, however, until the listener is started again.

End TCP/IP Server command restrictions

If the End TCP/IP Server command is used to end the DDM listener when it is not active, a diagnostic message will be issued. This same diagnostic message will not be sent if the listener is not active when an ENDTCPSVR SERVER(*ALL) command is issued.

Examples: End TCP/IP Server (ENDTCPSVR) command

These examples demonstrate using the End TCP/IP server (ENDTCPSVR) CL command.

This command ends all TCP/IP servers.

```
ENDTCPSVR *ALL
```

This command ends the DDM server.

```
ENDTCPSVR SERVER(*DDM)
```

Start DDM listener in iSeries Navigator:

The DDM listener can also be administered using iSeries Navigator, which is part of iSeries Access Family.

This can be done by following this path: **Network → Servers → TCP/IP directory**.

DDM server jobs

This topic collection provides information for working with server jobs.

Subsystem descriptions and prestart job entries with DDM:

A subsystem description defines how, where, and how much work enters a subsystem, and which resources the subsystem uses to perform the work.

The following paragraphs describe how the prestart job entries in the QUSRWRK subsystem description affect the DDM server.

A prestart job is a batch job that starts running before a program on a remote server initiates communications with the server. Prestart jobs use prestart job entries in the subsystem description to determine which program, class, and storage pool to use when the jobs are started. Within a prestart job entry, you must specify attributes that the subsystem uses to create and manage a pool of prestart jobs.

Prestart jobs provide increased performance when initiating a connection to a server. Prestart job entries are defined within a subsystem. Prestart jobs become active when that subsystem is started, or they can be controlled with the Start Prestart Job (STRPJ) and End Prestart Job (ENDPJ) commands.

DDM prestart jobs:

System information that pertains to prestart jobs (such as DSPACTPJ) will use the term *program start request* exclusively to indicate requests made to start prestart jobs, even though the information might pertain to a prestart job that was started as a result of a TCP/IP connection request.

The following list contains the prestart job entry attributes with the initially configured value for the DDM TCP/IP server. They can be changed with the Change Prestart Job Entry (CHGPJE) command.

- Subsystem description. The subsystem that contains the prestart job entries is QUSRWRK.
- Program library and name. The program that is called when the prestart job is started is QSYS/QRWTSRVR.
- User profile. The user profile that the job runs under is QUSER. This is what the job shows as the user profile. When a request to connect to the server is received from a client, the prestart job function swaps to the user profile that is received in that request.
- Job name. The name of the job when it is started is QRWTSRVR.
- Job description. The job description used for the prestart job is *USRPRF. Note that the user profile is QUSER so this will be whatever QUSER's job description is. However, the attributes of the job are changed to correspond to the requesting user's job description after the user ID and password (if present) are verified.
- Start jobs. This indicates whether prestart jobs are to automatically start when the subsystem is started. These prestart job entries are shipped with a start jobs value of *YES. You can change these to *NO if the DDM TCP/IP communications support is to be used.

  Note: If the DDM server jobs are not running and the DDM listener job is batch, immediate DDM server jobs will still be run under the QSYSWRK subsystem.
- Initial number of jobs. As initially configured, the number of jobs that are started when the subsystem is started is 1. This value can be adjusted to suit your particular environment and needs.
- Threshold. The minimum number of available prestart jobs for a prestart job entry is set to 1. When this threshold is reached, additional prestart jobs are automatically started. This is used to maintain a certain number of jobs in the pool.
- Additional number of jobs. The number of additional prestart jobs that are started when the threshold is reached is initially configured at 2.

- Maximum number of jobs. The maximum number of prestart jobs that can be active for this entry is *NOMAX.
- Maximum number of uses. The maximum number of uses of the job is set to 200. This value indicates that the prestart job will end after 200 requests to start the server have been processed. In certain situations, you might need to set the MAXUSE parameter to 1 in order for the TCP/IP server to function properly. When the server runs certain ILE stored procedures, pointers to destroyed objects might remain in the prestart job environment; subsequent uses of the prestart job would cause MCH3402 exceptions.
- Wait for job. The *YES setting causes a client connection request to wait for an available server job if the maximum number of jobs is reached.
- Pool identifier. The subsystem pool identifier in which this prestart job runs is set to 1.
- Class. The name and library of the class the prestart jobs will run under is set to QSYS/QSYSCLS20.

When the start jobs value for the prestart job entry has been set to *YES, and the remaining values are as provided with their initial settings, the following things happen for each prestart job entry:
- When the subsystem is started, one prestart job is started.
- When the first client connection request is processed for the TCP/IP server, the initial job is used and the threshold is exceeded.
- Additional jobs are started for the server based on the number defined in the prestart job entry.
- The number of available jobs will not reach below 1.
- The subsystem periodically checks the number of prestart jobs in a pool that are unused and ends excess jobs. It always leaves at least the number of prestart jobs specified in the initial jobs parameter.

Related tasks

"Configure the DDM server job subsystem" on page 130
By default, the DDM TCP/IP server jobs run in the QUSRWRK subsystem. Using iSeries Navigator, you can configure DDM server jobs to run all or certain server jobs in alternate subsystems based on the client's IP address.

Monitor prestart jobs:

Prestart jobs can be monitored by using the Display Active Prestart Jobs (DSPACTPJ) command.

The DSPACTPJ command provides the following information:
- Current number of prestart jobs
- Average number of prestart jobs
- Peak number of prestart jobs
- Current number of prestart jobs in use
- Average number of prestart jobs in use
- Peak number of prestart jobs in use
- Current number of waiting connect requests
- Average number of waiting connect requests
- Peak number of waiting connect requests
- Average wait time
- Number of connect requests accepted
- Number of connect requests rejected

Manage prestart jobs:

The key is to ensure that there is an available prestart job for every sent request that starts a server job.

The information presented for an active prestart job can be refreshed by pressing the F5 key on the Display Active Prestart Jobs display. Of particular interest is the information about program start requests. This information can indicate to you whether you need to change the available number of prestart jobs. If you have information indicating that program start requests are waiting for an available prestart job, you can change prestart jobs using the Change Prestart Job Entry (CHGPJE) command.

If the program start requests were not being acted on fast enough, you can do any combination of the following things:

- Increase the threshold.
- Increase the Initial number of jobs (INLJOBS) parameter value.
- Increase the Additional number of jobs (ADLJOBS) parameter value.

Remove prestart job entries:

If you decide that you do not want the servers to use the prestart job function, you must follow these steps.

1. End the prestarted jobs using the End Prestart Job (ENDPJ) command.

   Prestarted jobs ended with the ENDPJ command will be started the next time the subsystem is started if start jobs *YES is specified in the prestart job entry or when the STRHOSTSVR command is issued for the specified server type. If you only end the prestart job and do not perform the next step, any requests to start the particular server will fail.
2. Remove the prestart job entries in the subsystem description using the Remove Prestart Job Entry (RMVPJE) command.

   The prestart job entries removed with the RMVPJE command are permanently removed from the subsystem description. After the entry is removed, new requests for the server will be successful, but will incur the performance overhead of job initiation.

Routing entries:

An i5/OS job is routed to a subsystem by using the routing entries in the subsystem description.

The routing entry for the listener job in the QSYSWRK subsystem is present after i5/OS is installed. This job is started under the QUSER user profile, and the QSYSNOMAX job queue is used.

The server jobs run in the QUSRWRK subsystem also. The characteristics of the server jobs are taken from their prestart job entry which also comes automatically configured with i5/OS. If this entry is removed so that prestart jobs are not used for the servers, then the server jobs are started using the characteristics of their corresponding listener job.

The following list provides the initial configuration in the QSYSWRK subsystem for the listener job.

Subsystem
QSYSWRK

Job Queue
QSYSNOMAX

User QUSER

Routing Data
QRWTLSTN

Job Name
QRWTLSTN

Class QSYSCLS20

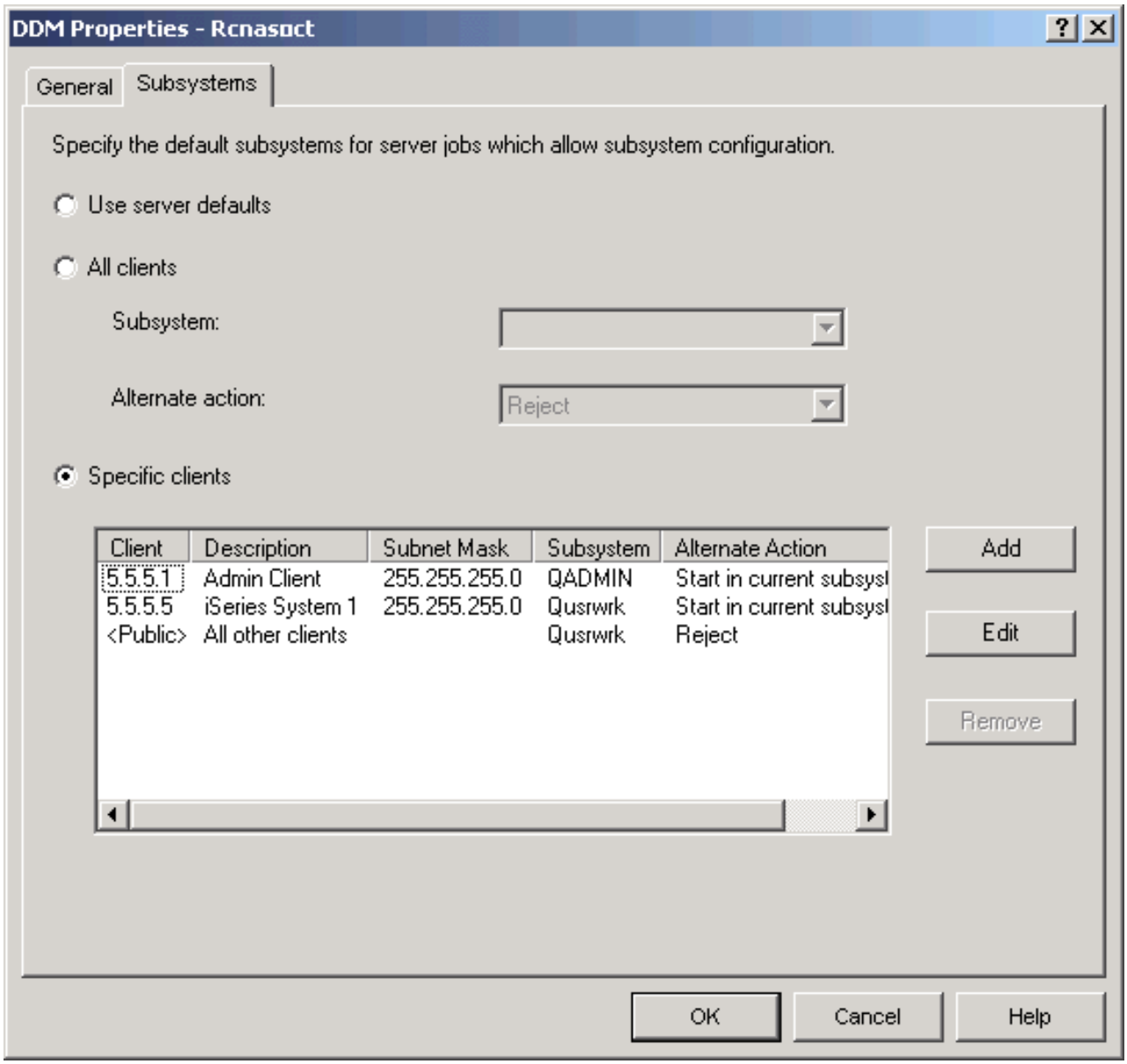
Configure the DDM server job subsystem

By default, the DDM TCP/IP server jobs run in the QUSRWRK subsystem. Using iSeries Navigator, you can configure DDM server jobs to run all or certain server jobs in alternate subsystems based on the client's IP address.

To set up the configuration, follow these steps:

1. Create a prestart job entry for each desired subsystem with the ADDPJE CL command.
2. Start the prestart job entry you created with the STRPJ CL command.
3. In iSeries Navigator, expand **Network**.
4. Expand **Servers**.
5. Click **TCP/IP**.
6. Right-click **DDM** in the list of servers that are displayed in the right panel and select **Properties**.
7. On the **Subsystems** tab, add the specific client and the name of the subsystems.

In the following example, the administrator can connect and run in the QADMIN subsystem, while another server in the network can connect and run in QUSRWRK. All other clients would be rejected.



Related reference

"DDM prestart jobs" on page 127
System information that pertains to prestart jobs (such as DSPACTPJ) will use the term *program start request* exclusively to indicate requests made to start prestart jobs, even though the information might pertain to a prestart job that was started as a result of a TCP/IP connection request.

Identify server jobs

If you look at the server jobs started on the system, you might find it difficult to relate a server job to a certain application requester job or to a particular PC client. Being able to identify a particular job is a prerequisite to investigating problems and gathering performance data. iSeries Navigator provides support for these tasks that make the job much easier.

iSeries job names:

The job name used on the iSeries consists of three parts: the simple job name, user ID, and job number (ascending order).

The DDM server jobs follow the following conventions:

- Job name is QRWTSRVR.
- User ID
  - Will always be QUSER, whether prestart jobs are used or not.
  - The job log will show which user is currently using the job.
- The job number is created by work management.

Display server jobs:

There are several methods that can be used to aid in identifying server jobs. One method is to use the WRKACTJOB command. Another method is to use the WRKUSRJOB command. A third method is to display the history log to determine which job is being used by which client user.

Display active jobs using the WRKACTJOB command

The WRKACTJOB command shows all active jobs. All server jobs are displayed, as well as the listener job.

The following figures show a sample status using the WRKACTJOB command. Only jobs related to the server are shown in these figures. You must press F14 to see the available prestart jobs.

The following types of jobs are shown in the figures.

- 1 - Listener job
- 2 - Prestarted server jobs

```
                        Work with Active Jobs                AS400597
                                                    04/25/97  10:25:40
CPU %:   3.1   Elapsed time:  21:38:40  Active jobs:  77

Type options, press Enter.
  2=Change   3=Hold   4=End   5=Work with   6=Release   7=Display message
  8=Work with spooled files   13=Disconnect ...

Opt  Subsystem/Job  User     Type  CPU %  Function          Status
            .
___   QUSRWRK       QSYS       SBS      .0                  DEQW
            .
___ 1
       QRWTLSTN     QUSER      BCH      .0                  SELW
           .
           .
___ 2
       QRWTSRVR     QUSER      PJ       .0                  TIMW
___    QRWTSRVR     QUSER      PJ       .0                  TIMW
___    QRWTSRVR     QUSER      PJ       .0                  TIMW
___    QRWTSRVR     QUSER      PJ       .0                  TIMW
___    QRWTSRVR     QUSER      PJ       .0                  TIMW
           .                                                          More...
```

The following types of jobs are shown:

PJ The prestarted server jobs.

SBS The subsystem monitor jobs.

BCH The listener job.

Display active user jobs using the WRKUSRJOB command

The command WRKUSRJOB USER(QUSER) STATUS(*ACTIVE) will display all active server jobs running under QUSER. This includes the DDM listener and all DDM server jobs. This command might be preferable, in that it will list fewer jobs for you to look through to find the DDM-related ones.

Display the history log:

Each time a client user establishes a successful connection with a server job, that job is swapped to run under the profile of that client user. To determine which job is associated with a particular client user, you can display the history log using the DSPLOG command.

An example of the information provided is shown in the following figure.

```
                    Display History Log Contents
      .
      .
DDM job 036995/QUSER/QRWTSRVR servicing user MEL on 08/18/97 at 15:26:43.
      .
DDM job 036995/QUSER/QRWTSRVR servicing user REBECCA on 08/18/97 at 15:45:08.
      .
DDM job 036995/QUSER/QRWTSRVR servicing user NANCY on 08/18/97 at 15:56:21.
      .
DDM job 036995/QUSER/QRWTSRVR servicing user ROD on 08/18/97 at 16:02:59.
      .
DDM job 036995/QUSER/QRWTSRVR servicing user SMITH on 08/18/97 at 16:48:13.
      .
DDM job 036995/QUSER/QRWTSRVR servicing user DAVID on 08/18/97 at 17:10:27.
      .
      .
      .


Press Enter to continue.

F3=Exit   F10=Display all   F12=Cancel
```

Cancel distributed data management work

Whether you are testing an application, handling a user problem, or monitoring a particular device, there are times when you may want to end work that is being done on a server.

When you are using an interactive job, you normally end the job by signing off the server. There are other ways that you can cancel or discontinue jobs on the server. They depend on what kind of a job it is and what server it is on.

End Job (ENDJOB) command

The End Job (ENDJOB) command ends any job.

The job can be active, on a job queue, or already ended. You can end a job immediately or by specifying a time interval so that end of job processing can occur.

Ending a source job ends the job on both the source and the target. If the application is under commitment control, all uncommitted changes are rolled back.

End Request (ENDRQS) command

The End Request (ENDRQS) command cancels a local or source operation (request) that is currently stopped at a breakpoint.

This means the command cancels an application requester operation or request. You can cancel a request by entering ENDRQS on a command line or you can select option 2 from the System Request menu.

If it cannot be processed immediately because a server function that cannot be interrupted is currently running, the command waits until interruption is allowed.

When a request is ended, an escape message is sent to the request processing program that is currently called at the request level being canceled. Request processing programs can monitor for the escape message so that cleanup processing can be done when a request is canceled. The static storage and open files are reclaimed for any program that was called by the request processing program. None of the programs called by the request processing program are notified of the cancellation, so they have no opportunity to stop processing.

Attention: Using the ENDRQS command on a source job may produce unpredictable results and can cause the loss of the connection to the target.

Performance considerations for DDM

These topics provide information to help you improve performance when using DDM and also provide some information about when to use something other than DDM to accomplish some functions.

- When a DDM file is specified on the CPYF command, optimal performance is obtained if the following items are all true:
  - The from-file is a logical or physical file and the to-file is a physical file.
  - FMTOPT is *NONE, *NOCHK, or not specified.
  - INCHAR, INCREL, ERRLVL, RCDDMT (*ALL), PRINT(*COPIED), PRINT(*EXCLD), SRCSEQ, TOKEY, SRCOPT, or FROMKEY parameter is not specified.
  - The from-file is not overridden with the POS keyword, other than *NONE or *START.
  - The to-file is not overridden with INHWRT(*YES).
- The Open Query File (OPNQRYF) command uses System/38 extensions to the DDM architecture. The System/38 DDM architecture extensions minimize DDM system processing time. These extensions are not used when:
  - The source server is neither a System/38 nor an iSeries server
  - The target server is neither a System/38 nor an iSeries server
- You can greatly reduce the amount of data transferred between servers if you use query functions such as the iSeries command OPNQRYF OPTIMIZE(*YES). However, for user-written applications, the amount of data exchanged between the servers is larger than that used to communicate using DDM with non-iSeries servers. The additional data provides iSeries extended DDM functions and also reduces source server DDM processing overhead. Using normal read, write, update, add, and delete operations as examples, consider the following items:
  - Standard DDM architecture DDM overhead data includes such information as a file identification, operation code, and simple result information. A user program read-by-key operation uses approximately 40 characters of DDM information in addition to the length of the key data. Data returned from the remote server uses approximately 32 characters of DDM information plus the length of the data file record.
  - System/38 DDM architecture extensions cause additional data overhead such as record format identification and a major portion of the I/O feedback area information. A user program read-by-key operation uses approximately 60 characters of DDM information in addition to the length of the key data. Data returned from the remote server uses approximately 80 characters of DDM information plus the length of the data file record. Normally, the additional length in data streams is not noticeable. However, as line activity increases, line utilization might peak sooner when using these extended data streams as opposed to standard DDM data streams.
- The target DDM job priority is controlled by the job class specified by the associated subsystem description routing entry. The following routing entry is normally the one used for all target (program start request) jobs:

  ```
  ADDRTGE ... PGM(*RTGDTA) ... CMPVAL(PGMEVOKE 29)
  ```

  The subsystems QBASE and QCMN, which are shipped with the iSeries server, have this routing entry.

To have target DDM jobs in a subsystem run at a different priority than other APPC target jobs in the same subsystem, insert the following routing entry with the appropriate sequence number:

```
ADDRTGE SBSD(xxx) SEQNBR(nnn) CMPVAL(QCNTEDDM 37)
        PGM(*RTGDTA) CLS(uuu)
```

The class *uuu* is used to determine target job priority.

- To display records in a remote file, display station pass-through should be used whenever possible. Otherwise, direct record positioning should be used with the Display Physical File Member (DSPPFM) command.
- If a DDM user exit security program is a CL program and it creates an i5/OS exception and an inquiry message that requires the target system operator to respond, both the user exit program and the source server job must wait for the response. Consider using the default system reply list by specifying INQMSGRPY(*SYSRPYL) for the TDDM job's description specified on the Add Communications Entry (ADDCMNE) command for that APPC remote location information.
- The WAIT and WAITFILE parameters, used on commands like Allocate Object (ALCOBJ) or Receive Message (RCVMSG), have no effect on the source server program. These parameters function the same as they do when local files are accessed. The wait time values specified on commands run on the source server do not take effect on the source system; they affect only the target server and only if it is an iSeries server or a System/38.

  Notes:

  1. The WAITFILE parameter of the create or change i5/OS-Intersystems Communications Function (ICF) file command determines how long the APPC support will wait for session resources to become available when doing an acquire operation or a start function. The WAITFILE value is not used for sessions where the connection to the adjacent server is over a switched connection. An example is an SDLC switched line, an X.25 SVC line, an Ethernet line, or a token-ring connection. Using a switched connection, acquire operations and start functions do not time out.
  2. Because APPN sessions might cross multiple servers and lines to reach the remote server, the WAITFILE timer should be adjusted to allow more time in these cases. You should not specify *IMMED for the WAITFILE parameter if your application is running in a network configured to use APPN functions. The value you specify for this parameter is dependent on the size and type of the network.
- As for any LU session type 6.2 data exchange, certain SNA parameters can affect performance. These parameters include the path information unit size (MAXFRAME), the request/response unit size (MAXLENRU), SNA pacing (INPACING, OUTPACING), and for X.25, packet size and window size. In general, the larger the value used, the better the realized performance.
- **SNA path information unit size**

  The path information unit (PIU) is the size of the actual data transmission block between two servers. The MAXFRAME parameter on the Create Controller Description (APPC) (CRTCTLAPPC) or Create Controller Description (SNA Host) (CRTCTLHOST) command specifies the path information unit size the local server attempts to use. During session establishment, both servers determine which size is used, and it is always the smaller value. Other remote servers might have different PIU size considerations.
- **SNA response/request unit size**

  The response/request unit (RU) size (CRTMODD MAXLENRU) controls the amount of server buffering before fitting that data into the path information unit that is actually transmitted. In APPC, the transmit and receive RU lengths are negotiated during session establishment. Again, the negotiation results in the smallest value being used. Other remote servers have different RU size considerations.
- **SNA pacing values**

  The pacing value determines how many request/response units (RUs) can be received or sent before a response is required indicating buffer storage is available for more transmissions. During session establishment, both servers determine which size is used, and it is always the smaller value.

In cases where both batch and interactive processing occur at the same time on the same communications line, iSeries job priority might be used to favor interactive processing over batch processing. In addition, reducing the value of pacing for a batch application and raising it for an interactive application might be necessary to provide a level of line activity priority for the interactive application.

On an iSeries server, different pacing values can be specified through the creation of different MODES (Create Mode Description [CRTMODD] command) to the different applications. Other remote systems have different SNA pacing value considerations.

- **X.25 packet**

  An X.25 packet smaller than the MAXFRAME value adds data transmission time over a non-X.25 data link. In general, for X.25, the longer the MAXFRAME and the actual amount of data being transmitted, the greater this difference is. In the case of DDM, which adds DDM control information to the normal file record data, the packet size has an additional effect on the difference between local and remote file processing and between non-X.25 and X.25 data links.

  In cases of many deblocked DDM operations, the number of packets required to transmit data might become so high that packet processing overhead within the X.25 adapter affects performance significantly. Use the largest X.25 packet window size supported by the network and communicating products to maximize performance.

  When X.25 must be used to access remote files, successive transmission of many small deblocked records, such as less than 80 character records, might cause the X.25 adapter to expend a disproportionate amount of time processing X.25 packet characters as opposed to transmission of user data.

  In general, the overhead in processing X.25 packets results in less throughput than the use of a conventional line when identical line speeds are used and data transfer is in only one direction. When data is transferred at the same time in both directions, the advantages of X.25 duplex support is realized. On the System/38, the overall processing effect is minimal, because the overhead in processing the packets is done within the Integrated X.25 Adapter.

In general, the processing of remote files by using DDM is transparent to an application program or utility function, such as that provided by the Copy File (CPYF) command. However, additional time is required when accessing remote files by using a communications line. The performance difference between local file and remote file processing is proportional to the number of accesses to remote files, the data record length, and the line speed during a unit of performance measurement.

An additional difference between local and remote file processing is that the input or output operation to a local file might not result in an immediate physical disk operation because the server transfers blocks of data from the disk and writes blocks of data to the disk. There are times, then, that the user program accesses data within main storage and the physical I/O occurs at a different time. Therefore, to minimize the difference between local file and remote file performance, it is essential that knowledge of an application design and the amount and type of accesses to files be considered when determining which files are to be accessed remotely using DDM.

The additional time for each remote access is comprised of:

- Additional system processing to convert local server file interfaces to the DDM architecture interfaces
- Amount of data transmitted over the communications line
- Amount of remote system processing of the file operations
- Speed of the communications line

The communications line time accounts for most of the additional time, though the actual time is dependent on line speed and the amount of line activity during the DDM function.

As is true in non-DDM cases, local and remote server job priorities have the most significant effect on performance. On an iSeries server, the PRIORITY and TIME SLICE values of the class being used control

job priority. The SDDM runs under the source job, and the TDDM runs under the class assigned to the APPC routing entry of the target server's subsystem. In applications that access multiple files, the best results are achieved when the most heavily accessed files are on the same server as the program that is running and the less heavily accessed files are on a remote server. Key considerations regarding the placement of files and application programs follow:

- The system having primary responsibility for file maintenance needs to be identified. In all cases of multiple servers applications, the best performance results if only one server is responsible for file maintenance. If an application program maintains the file through exclusive (nonshared) processing, best performance can be realized when the application program resides on the system with the file.

  In some cases, transmitting the file back to the local server might require:

  - An APPC program.
  - A program using remote DDM files.
  - The Copy File (CPYF) command by using DDM.
  - Object distribution SNDNETF and RCVNETF operations. In interactive applications, display station pass-through should be considered when the amount of display data transferred is significantly less than the amount of database file data that would be sent by using DDM.
- In cases where file placement requires movement of application processing to a remote server for best performance results, use of the Submit Remote Command (SBMRMTCMD) command should be considered. This works best in a batch processing input stream where each program waits for the preceding program to complete. The use of the SBMRMTCMD command is valid only when the source and target servers are iSeries servers or Systems/38s. For example, assume that program A accesses local files. Program A would run on a local server. Program B accesses remote files. You can use the SBMRMTCMD command to run program B on the remote server.
- In cases where file maintenance is shared across servers, the best performance can be obtained if the file is placed on the server with the largest percentage of file update, add, and delete operations.

  In certain cases, a pair of source and target APPC programs can provide performance improvements over DDM. For example, assume 10 records are to be retrieved from the remote server. When DDM is used and record blocking cannot be used (for example, user program random input operation, sequential input for change, or use of the OVRDBF SEQONLY[*NO] command), assume 10 data transmissions are sent and 10 are received, for a total of 20 transmissions. User-written APPC programs can build additional intelligence into the data stream such that request for the data and receipt of the data can be done in two data transmissions instead of 20, one request for all records of customer 00010 and one response containing 10 records for customer 00010.

Consider two sample application processing techniques, batch file processing and interactive file processing.

Related concepts

"Security" on page 48
This topic describes how iSeries security relates to DDM, and how it can limit access to the data resources of a target server by source server programs and users.

 Communications Management PDF

APPC, APPN, and HPR

 LAN, Frame-Relay, and ATM Support PDF

Related reference

"Display DDM remote file records" on page 122
The Display Physical File Member (DSPPFM) command can be used to display a remote file on a target server.

"User exit program considerations for DDM" on page 66
There are some considerations that you should understand before using user exit programs for DDM.

Batch file processing with DDM

Consider these items when using batch file processing with DDM.

- When an application opens a local file for *sequential input only* or *output add*, the server uses blocking techniques to achieve maximum throughput. To ensure blocking is used for a remote file accessed by using DDM, do not use random record processing operations in the program but specify OVRDBF SEQONLY(*YES) against the DDM files opened by the program.
- Use of read and read-next operations in the high-level language (HLL) program to access the file maximizes the effect of the SEQONLY(*YES) specification.
- The use of random processing operations, such as chain operations of ILE RPG or start operations of ILE COBOL programming language, causes DDM to send deblocked operations across the communications line even if the application processes the file data sequentially. This results in significant differences between local and remote file processing.
- When simple physical file transfer is desired (all records transferred and no application processing of the data), use of DDM by using the Copy File (CPYF) command, or a user-written program using DDM with the Override Database File (OVRDBF) command SEQONLY(*YES number-of-records) specified, transfers the data more quickly than a user-written APPC program. The Copy File command and the DDM SEQONLY(*YES) support require less calls and returns between the program and APPC data management modules than does a standard ILE RPG or ILE COBOL APPC program.
- For ILE RPG or ILE COBOL sequential input-only applications, SEQONLY(*YES) should be specified with no *number of records* to achieve best throughput. For ILE RPG or ILE COBOL sequential output-only applications to keyed files, a large *number-of-records* value should be used.
- The Send Network File (SNDNETF) command can be considered as an alternative to DDM or user-written APPC programs when transferring all records within a file to a remote iSeries server. The SNDNETF command requires SNADS to be configured on the source and target iSeries server. If one or more intermediate servers are between the source and target iSeries servers, SNADS provides intermediate node routing of the data when correctly configured.
- Use of the SNDNETF command by using SNADS offers the advantages of transmitting one copy of the data to multiple users on one or more target servers through a multiple node network, and the time scheduled transmission of that data by using the SNADS distribution queue parameter.

  However, in addition to requiring SNADS to use the SNDNETF command, the target server user must also run the Receive Network File (RCVNETF) command to make the file usable on the target server. Use of DDM would not require this additional target server processing.

  In general, the file transmission times by using SNADS (user program DDM sequential file processing, the DDM Copy File command, and a user-written APPC program between two iSeries servers) are within 10% of each other. However, the use of the SNDNETF and RCVNETF commands to make a copy of the data usable on the target server does add total processing time over the other methods of file transfer.
- Because the SNDNETF command can transmit objects within a save file, the amount of data that is actually sent by using this technique might be less than that sent using the other techniques. If the database file data sent contains a significant number of duplicate character strings, use of the Save Object (SAVOBJ) command parameter DTACPR(*YES) (data compression) can significantly reduce the amount of data that is actually sent by using a SNADS distribution. However, if there are few duplicate character strings, there is little change in the amount of data sent.
- The iSeries file transfer subroutines might also be used to transfer an entire file between iSeries servers and an iSeries server and a System/36. These subroutines might be called from high-level language programs, and in some cases throughput is achieved similar to that by using DDM.

See the *SNA Distribution Services* manual on the V5R1 Supplemental Manuals Web site.

Related concepts

Communications Management PDF

Related reference

"Use of object distribution" on page 123
Although DDM file names can be specified on the Send Network File (SNDNETF) and Receive Network File (RCVNETF) commands, these commands should be run, whenever possible, on the server where the data actually exists. Therefore, if both servers are iSeries servers and both are part of a SNADS network, *object distribution* can be used instead of DDM to transfer the data between them.

Related information

ICF Programming PDF

Interactive file processing with DDM

Consider these items when using interactive file processing with DDM.

- The greater the number of random file operations per unit of performance measurement, the greater the difference between local and remote file processing because each operation has to be sent separately across the communications line. DDM cannot anticipate the next operation.

  Using a simple inquiry application that produces display output, by using workstation subfile support (as an example), consider an application that does two random record retrievals per Enter key versus one that does 15 random record retrievals. The operator might barely notice a delay in response time when two records are retrieved. However, there would be a noticeable difference between local and remote response time when 15 records are retrieved randomly from the remote server.
- Use of display station pass-through should be considered when the amount of data transferred back to the local (source) server per unit of performance measurement significantly exceeds the amount of data presented on the display. Test results have shown that the total elapsed time between a single deblocked DDM get record operation and an equivalent user-written APPC operation is very close, with APPC being slightly quicker. The DDM operation does require more processing seconds than the direct APPC interface.

  Also, because each DDM operation always requires an operation result response from the remote server to ensure data integrity, user-designed partner APPC programs can offer an advantage for update, add, and delete operations by not validating the result of the operation until a later time.
- Be aware that additional time is needed when accessing files on other servers, particularly the time required for communications over the line. This should be considered when determining whether the file should be a local or remote file, especially if it is to be used often.

DDM conversation length considerations

Consider these items regarding the length of conversations when using DDM.

- Within a source job, if it is likely that a DDM conversation will be used by more than one program or DDM file, *KEEP is the value that should be specified for the DDMCNV job attribute. This saves the time and resources needed to start a target job (TDDM) each time a DDM file is accessed for the same location and mode combination within the source job.
- There is significant server and communications line overhead when a target DDM manager is started. The processing includes the APPC program start request, server type identification, and file open processing. However, if it is not necessary to keep the conversation active, *DROP should be specified for DDMCNV. When the local DDM file is closed, the session being used is released for use by other jobs using DDM or other APPC functions, such as SNADS and display station pass-through, to the same remote server.
- When the source and target servers are iSeries servers or System/38, the file input and output requests made by an application program use a form of DDM support that minimizes the amount of time needed to code and decode each request. This is accomplished by System/38 extensions to the DDM architecture.

  When the source and target servers are neither an iSeries server nor a System/38, then System/38 extensions to the DDM architecture are not used.

DDM problem analysis on the remote server

Some functions that involve a target server may take a relatively long period of time to complete. In these situations, the target server may not appear to be functioning when it is actually waiting for a reply.

Any messages created on the target server (such as file full) are sent to the system operator's message queue on the target server. (All DDM-related messages are logged in the target server's job log.) In most cases, a message similar to the one sent to the target system operator is also sent to the source server (with a different message number), but only after the target system operator has replied to the message.

If no job log is found on the target server, the Submit Remote Command (SBMRMTCMD) command can be used to send a Change Job Description (CHGJOBD) command to the target server to change the message logging level.

Another consideration is when end-of-file delay is being used between two iSeries servers, canceling the job on the source server does not cancel the job on the target server. Or, if the source system job is canceled while the target job is performing some function, the target job is not canceled.

In some situations, it may be necessary for a user on either the source or target server to call the other location or use pass-through to determine the status of the job on that end and to reply to any messages waiting for a response.

Handle connection request failures for TCP/IP

The main causes for failed connection requests at a server configured for TCP/IP use is that the DDM TCP/IP server is not started, an authorization error occurred, or the machine is not running.

DDM server is not started or the port ID is not valid:

The error message given if the DDM TCP/IP server is not started is CPE3425.

The message text is:

```
A remote host refused an attempted connect operation.
```

You can also get this message if you specify the wrong port on the Create DDM file (CRTDDMF) or Change DDM File (CHGDDMF) command. For a DB2 UDB for iSeries server, not using the secure sockets protocol, the port should always be 446 or 447. It is recommended that the 446 always be used for clear text transmissions, and 447 be used for Internet Protocol Security Architecture (IPSec). To start the DDM server on the remote server, run the STRTCPSVR *DDM command. You can request that it be started whenever TCP/IP is started by running the CHGDDMTCPA AUTOSTART(*YES) command.

DDM connection authorization failure:

The error messages given for an authorization failure is CPF9190.

The message text is:

```
Authorization failure on DDM TCP/IP connection attempt.
```

The cause section of the message gives a reason code and a list of meanings for the possible reason codes. Reason code 17 means that there was an unsupported security mechanism (SECMEC).

Prior to V4R5, there were two SECMECs implemented by DB2 UDB for iSeries that an iSeries application requester could use: user ID only and user ID with password. In V4R5, support was added for the encrypted password security mechanism. However, the encrypted password will be sent only if a password is available at the time the connection is initiated.

The default required SECMEC for an iSeries server is user ID with password. If the source server sends only a user ID to a server with the default SECMEC, the above error message with reason code 17 is given.

Solutions for the unsupported SECMEC failure are:

1. To allow the user ID only SECMEC at the server by running the CHGDDMTCPA PWDRQD (*NO) command
2. To send at least a clear-text password on the connect request if PWDRQD (*YES) is in effect at the server
3. To send an encrypted password if PWDRQD (*ENCRYPTED) is in effect at the server

A password can be sent by using the ADDSVRAUTE command to add the remote user ID and password in a server authorization entry for the user profile under which the connection attempt is to be made.

An attempt will automatically be made to send the password encrypted in V4R5 and later systems. Note that pre-V4R5 iSeries servers cannot send encrypted passwords, nor can they decrypt encrypted passwords of the type sent by V4R5 iSeries servers.

Note that you have to have system value QRETSVRSEC (Retain Server Security Data) set to '1' to be able to store the remote password in the server authorization entry.

| **Attention:** For non-RDB DDM files, you must enter the RDB name of QDDMSERVER on the
| ADDSVRAUTE command in uppercase for use with DDM or the name will not be recognized during
| connect processing, and the information in the authorization entry will not be used.

DDM server not available:

If a remote server is not up and running, or if you specify an incorrect IP address or remote location name in the DDM file, you will get message CPE3447.

The message text is as follows:

```
A remote host did not respond within the timeout period.
```

There is normally a several minute delay before this message occurs. It might appear that something is hung up or looping during that time.

Not enough prestart jobs at server:

If the number of prestart jobs associated with the TCP/IP server is limited by the QRWTSRVR prestart job entry of the QUSRWRK or user-defined subsystem, and all prestart jobs are being used for a connection, an attempt at a new connection will fail with these messages.

CPE3426
A connection with a remote socket was reset by that socket.

CPD3E34
DDM TCP/IP communications error occurred on recv() — MSG_PEEK.

You can avoid this problem at the server by setting the MAXJOBS parameter of the CHGPJE command for the QRWTSRVR entry to a higher number or to *NOMAX, and by setting the ADLJOBS parameter to something other than 0.

System/36 source and target considerations for DDM

Before an iSeries server can access files on a System/36, Level 1.0 of the DDM architecture must be installed on the System/36. These topics contain information that applies when an iSeries server is the source or target server communicating with a System/36.

DDM-related differences between iSeries and System/36 files

Because of differences between the types of files supported by an iSeries server and a System/36, several items need to be considered when DDM is used between these two servers.

Generally, when a System/36 file is created locally (by the BLDFILE utility, for example), the System/36 user specifies such things as the type of file (S = sequential, D = direct, or I = indexed), whether records or blocks are to be allocated, how many of them are to be allocated, and how many additional times this amount can be added to the file to extend it.

Also, you can specify whether the file is to be *delete-capable* (DFILE) or not (NDFILE). In files specified as *not delete-capable*, records can be added or changed in the file, but not deleted.

Once these attributes have been specified, System/36 then creates the file and fills it with the appropriate hexadecimal characters. If a System/36 user specifies the file as:

- A *sequential* file, the entire file space is filled with hex 00 characters and the end-of-file (EOF) pointer is set to the beginning of the initial extent. If you attempt to read an empty sequential file, an EOF condition is received.
- A *direct* file that is *delete-capable*, the entire file space is filled with hex FF characters (deleted records) and the EOF pointer is set to the end of the initial extent. If you attempt to read an empty direct file that is delete-capable, a record-not-found condition is received.
- A *direct* file that is *not delete-capable*, the entire file space is filled with hex 40 characters (blank or null records) and the EOF pointer is set to the end of the initial extent. If you attempt to read an empty direct file that is not delete-capable, a blank record is returned for every record in the file until the end of the file is reached.
- An *indexed* file, it is prepared in the same manner as sequential files.

Typically, once a delete-capable file has been in use, it contains a relatively continuous set of active records with only a few deleted records, possibly an end of data marker, and then a continuous set of deleted records to the end of the file (EOF) space. This means that, unless the file is reorganized, a user can *un*delete (recover) a deleted record.

Of the three types of System/36 files, System/36 indexed files differ little from iSeries-supported logical files. If an iSeries source program is to use DDM to access the other types of files on a System/36, the iSeries application programmer should first consider the items remaining in this topic selection that relate to System/36.

System/36 source to iSeries target considerations for DDM

When System/36 is using DDM to communicate as a source server to access files on an iSeries target server, the information in this topic applies and should be considered.

- When System/36 creates a *direct* file on an iSeries server, the iSeries server creates a nonkeyed physical file with the maximum number of records, and prepares them as deleted records. The DDM architecture command Clear File (CLRFIL), when issued from a non-iSeries source server, clears and prepares the file; the CL command Clear Physical File Member (CLRPFM), when issued by a local or remote iSeries server, does not prepare the file.
- System/36 supports a maximum of three key definitions for logical files and one key definition for keyed physical files.
- Nondelete-capable direct files cannot be created using DDM on an iSeries server. In addition, the iSeries server does not support nondelete-capable files for all file organizations.

iSeries source to System/36 target considerations for DDM

When an iSeries server is using DDM to communicate as a source server to access files on a System/36 target server, the information in this topic applies and should be considered.

- Some file operations that are not rejected by a target iSeries server might be rejected by a target System/36. Examples are:

  - A delete record operation is rejected if the System/36 file is not a delete-capable file. To the iSeries source user, the rejection might appear to occur for unknown reasons.
  - Change operation that attempts to change the key in the primary index of a System/36 file is always rejected.
- In the System/36 environment, when System/36 users try to copy a delete-capable file to a file that is not delete-capable with the NOREORG parameter, a warning message is issued stating that deleted records might be copied. The user can choose option 0 (Continue) to continue the process. By selecting this option, the file is copied and any deleted records in the input file become active records in the output file. An iSeries server rejects the copy request if the user specifies COMPRESS(*NO).
- If data is copied to a target System/36 file that is a direct file and is not delete-capable, default values for all Copy File (CPYF) command parameters except FROMMBR, TOMBR, and MBROPT must be specified.
- An iSeries server does not support the overwriting of data on the Delete File (DLTF) command. If an iSeries user accessing a System/36 wants to overwrite the data, an application program must be written on the iSeries server, or the user must access the target System/36 and perform the overwrite operation.
- Depending on how a System/36 file is originally created, the maximum number of records it can contain is approximately eight million. This number can be significantly smaller if the file is not extendable or if sufficient storage space is not available to extend the file to add more records.
- System/36 supports a maximum of three key definitions for logical files and one key definition for keyed physical files.
- System/36 file support does not allow a file with active logical files to be cleared. When some iSeries programs (like ILE COBOL programs) open a file for output only, a command to clear the file is issued. A target System/36 rejects any such command to clear the file if a logical file exists over the file to be cleared.
- System/36 file support automatically skips deleted records. If an iSeries source user wishes to change the records in a System/36 base file over which at least one logical file has been built, the file must be opened in I/O mode, specifying direct organization and random access by record number. Then each record can be read by record number and changed. If a deleted record is found, a record-not-found indication is returned, and the record might be written rather than rewritten for that position (write operation rather than a change operation).
- System/36 file support also handles file extensions differently, depending on the file type and the language being used. However, an iSeries user cannot extend any type of System/36 file unless the access method used to access the file is similar to the method used when the file was created.

  If an iSeries user is accessing a System/36 file with an access method similar to the one used to create the file, the file can be extended during its use in the following manner:
  - If the file was created as a *sequential* file, the iSeries user should, if the iSeries language is:
    - ILE COBOL programming language: open the file using the EXTEND option.
    - PL/I: open the file using the UPDATE option. Perform a read operation using the POSITION option of LAST, and then perform the write operations.

  (BASIC and ILE RPG programming language both handle any needed file extensions automatically.)
- If the file was created as a *direct* file, the iSeries user should, if the iSeries language is:
  - ILE COBOL programming language: open the file using the I-O option, position the end of file pointer to the end of the file (using, for example, READ LAST), and perform a write operation.
  - PL/I: open the file using the UPDATE option, position the end of file (EOF) pointer to the end of the file (using, for example, READ LAST), and perform a write operation.

    (BASIC and ILE RPG programming language both handle any needed file extensions automatically.)
  - If the file was created as an *indexed* file, the file is extended each time a write operation is performed for a record having a key that does not already exist in the file.
- The iSeries user can access sequential System/36 files using either sequential or direct (by relative record number) methods, but significant differences occur when EOF or end of data occurs. If a

System/36 sequential file is being processed using relative record number access and is opened for either input/output or output only, then, on reaching the end of active records (EOF), you cannot add new records in the available free space beyond the end of data. You will have to close and reopen the file to extend it. To extend the file, you can either reopen it as a sequential file or open a logical file that uses this file as the base file.

- Because the normal access method used for a System/36 file can be changed by iSeries parameters to values other than *RMTFILE, it is possible that DDM might attempt to access the System/36 file in ways that the System/36 might not support. Normally, the default value (*RMTFILE) on the ACCMTH parameter gives the user the needed method of access. The use of access methods not normally expected (such as direct or sequential access to indexed files, or sequential access to direct files) requires the use of an ACCMTH parameter explicitly for the access.

  The normal access method used for a System/36 file can be changed on the iSeries server: by the ACCMTH parameter of the DDM file commands Create DDM File (CRTDDMF) and Change DDM File (CHGDDMF), by the SEQONLY parameter of the Override with Database File (OVRDBF) command, or by using the OVRDBF command to override one DDM file with another DDM file having a different ACCMTH value in effect.
- The iSeries user can access a System/36 file using a member name if the member name is *FIRST, or in some cases *LAST, or if the file name is the same as the member name.
- Target System/36 DDM cannot support creating logical files with duplicate (nonunique) keys, because the System/36 local data management key sort sends messages to the target server console with options 1 or 3 when duplicate keys are detected. This forces the target system operator either to change the file attributes to allow duplicate keys or to cancel the target data manager.

  Note: Never cancel the target data manager using a SYSLOG HALT.

Override considerations to System/36 for DDM

When a file override is issued on the iSeries server to get records in a logical file on a System/36, the results might be different than expected because of the difference in how each system deals with keyed files.

An iSeries server uses access paths and logical files, which produce a single view of a file. A System/36 logical file can be considered a list of keys and relative record numbers.

When an iSeries server accesses a System/36 logical file:

- If you specify a relative record number, you receive the record from the underlying System/36 base file that corresponds to that record number. Then if you request to read the next record, you receive the next sequential record from the base file.
- If you specify a key, you receive the record that corresponds to the first occurrence of that key in the index file. If you request to read the next record, you receive the record that matches the next entry in the index file.

The following example shows the various results for records being retrieved from a System/36 logical file by an iSeries program. The example assumes that:

- File S36FILEA is the base file and S36FILEB is the logical file that is built over the base file.
- Both files have DDM files named S36FILEA and S36FILEB that point to corresponding remote files on the target System/36.
- The key field is numeric and it always contains the record number.
- The records in the base file (S36FILEA) are in ascending sequence by key, and the records in the logical file (S36FILEB) are in descending sequence with the same key.
- To create the results shown in the following table, the POSITION parameter value is shown to vary, and no NBRRCDS parameter is specified on either command (which means the total records read is dependent only on the POSITION parameter value).

```
OVRDBF   FILE(S36FILEA)  TOFILE(S36FILEB)
   POSITION(*RRN ... or  *KEY ...)
CPYF   FROMFILE(S36FILEA)  TOFILE(ISERIESFILEB)
CRTFILE(*YES)  FMTOPT(*NOCHK)
```

Depending on the values specified on the Override with Database File (OVRDBF) command for the POSITION parameter, the following are the resulting records that are copied into the file ISERIESFILEB when it is created on the source iSeries server:

POSITION parameter (See note)	**Resulting records retrieved**
*RRN 1	299 records, 1 through 299
*KEY 1	1 record, first record only
*RRN 299	1 record, last record only
*KEY 299	299 records, 299 through 1
*RRN 150	150 records, 150 through 299
*KEY 150	150 records, 150 through 1
Note: This column assumes only one key field for *KEY values and uses the remote file name as the default value for the record format name.	

Personal computer source to iSeries target considerations for DDM

iSeries Access Family uses DDM to allow a personal computer to communicate as a source server to access objects on an iSeries target. iSeries Access Family uses Level 3.0 of the DDM architecture stream file access support to access folder management services (FMS) folders and documents.

The following considerations apply to iSeries Access Family use of the i5/OS DDM target support for the DDM architecture, Level 3.0. Other source servers that send Level 2.0 or Level 3.0 DDM architecture requests for stream files and directories may be able to use this information to help in connecting to an iSeries server by using DDM.

- A FMS must follow the file or directory name to access folder management services (FMS) folders and documents. There can be one or more blanks between the end of the name and the FMS.
- A leading slash (/) signifies the name is fully qualified. If there is no leading slash, any current directory in use is added to the front of the name given.
- The total length of a fully qualified document name is 76 characters. This includes any current directory that may be in use. This does not include the trailing FMS, which is used for typing purposes.
- A / FMS signifies the root folder for a directory name.
- To reduce the number of messages logged to the job log, some errors occurring on the iSeries target during open, get, put, and close document operations are not logged to the job log. See Table 9 for an illustration of these return codes.

Table 9. iSeries return codes

Description	**DDM reply**	**Function**
Folder not found	DRCNFNRM	OPEN
Folder in use	DRCIUSRM	OPEN
Document in use	FILIUSRM	OPEN
Document not found	FILNFNRM	OPEN
Document not found	EXSCNDRM	DELFIL
Document is read only	ACCINTRM	OPEN
End of data	SUBSTRRM	GET

Table 9. iSeries return codes (continued)

Description	DDM reply	Function
Data stream (DS) in use	STRIUSRM	GET
Data stream (DS) in use	STRIUSRM	PUT
Substring not valid	SUBSTRRM	UNLOCK
Unlocking a region that is not locked	EXSCNDRM	UNLOCK
File already open for the declare name	OPNCNFRM	OPEN
File not open	FILNOPRM	GET, PUT, LOCK, UNLOCK
Delete document SHDONL(TRUE) specified, but shadow does not exist	EXSCNDRM	DELFIL

- To provide better performance, the iSeries target handles the closing document in a manner such that when the document is closing, a command completion reply message (CMDCMPRM) is returned to the source server before the document is actually closed. If the document is damaged during the closing time, the user never receives this reply message unless he views the job log. When the user opens the file again, the updated data may not be there.
- An iSeries server does not support wait on the locking data stream function. The user on the source system must handle the wait function.

Examples: Code DDM-related tasks

The examples in this topic collection are based on representative application programs that might be used for processing data both on the local iSeries server and on one or more remote servers.

The first example is a simple inquiry application, and the second example is an order entry application. The third example accesses multiple files on multiple iSeries servers. The fourth example accesses multiple iSeries servers and a System/36.

The coding for each of these examples and tasks has one or two parts:

- Coding, shown in pseudocode form, not related to DDM but used to build the programming environment. The examples show you the task steps needed, independent of the language you use for your applications. You can write or adapt your programs in your language with the necessary coding to perform these or similar tasks.
- Coding, mostly done in CL, related to communicating with the other servers using DDM in the network.

References are made to other parts of this topic collection and to other topics for additional information that is helpful in understanding or using these examples.

Related reference

"Perform file management functions on remote servers" on page 119
i5/OS DDM supports creating, deleting, or renaming of files on a remote server.

Communications setup for DDM examples and tasks

These topics describe the network in which DDM is used for these task examples.

The network contains a central server in Philadelphia (an iSeries server), two remote iSeries servers in Toronto and New York City, a System/38 in Chicago, and a System/36 in Dallas. The Advanced Program-to-Program Communication (APPC) network for these servers was configured with the values shown in the following figure.

In this set of task examples, the System/36 has Release 5 of DDM installed and DDM with the compatible PTF installed. The System/38 has Release 8 of CPF installed with the DDM licensed program and the compatible program temporary fix (PTF) change applied to the server.

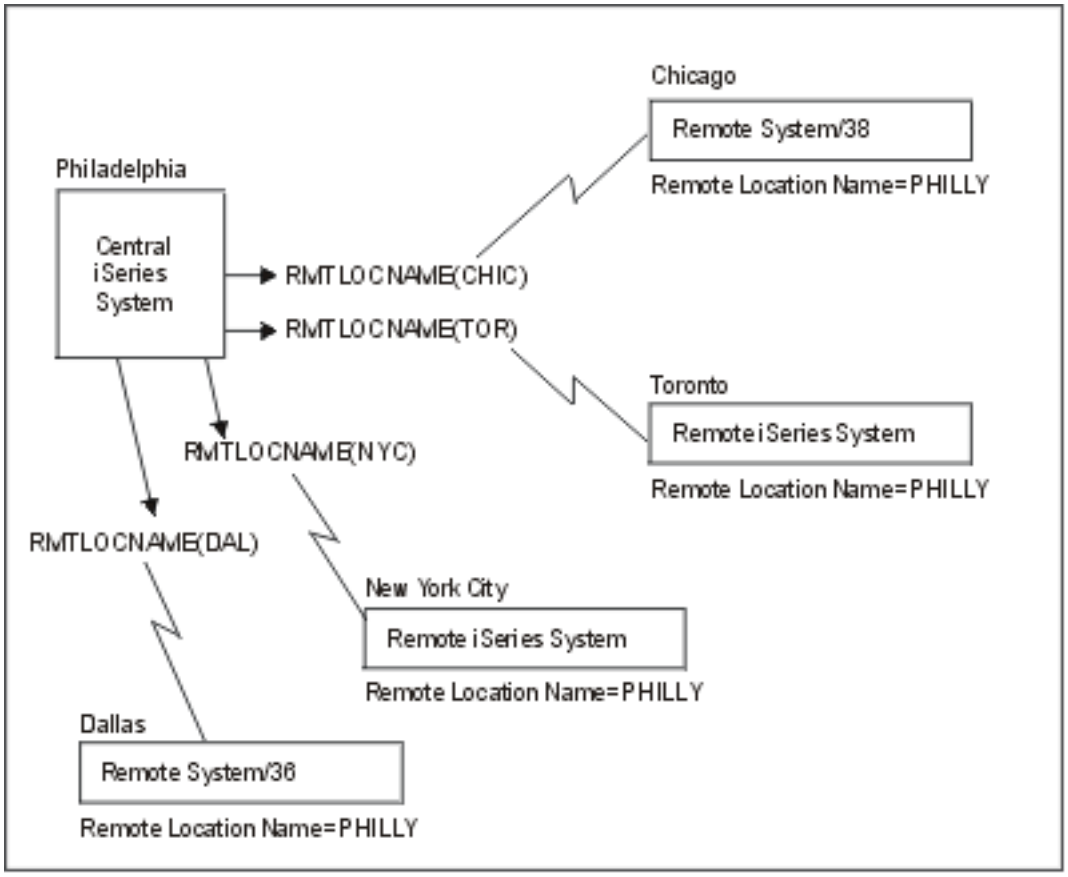


Figure 16. DDM network used in ORDERENT application tasks

DDM example 1: Simple inquiry application

This first example shows how multiple locations in a customer's business might be processing the same inquiry application on their own servers, using their own primary files. Without DDM, the two locations shown here (Chicago and Toronto) have their own primary file (CUSTMAST), both with different and duplicate levels of information.

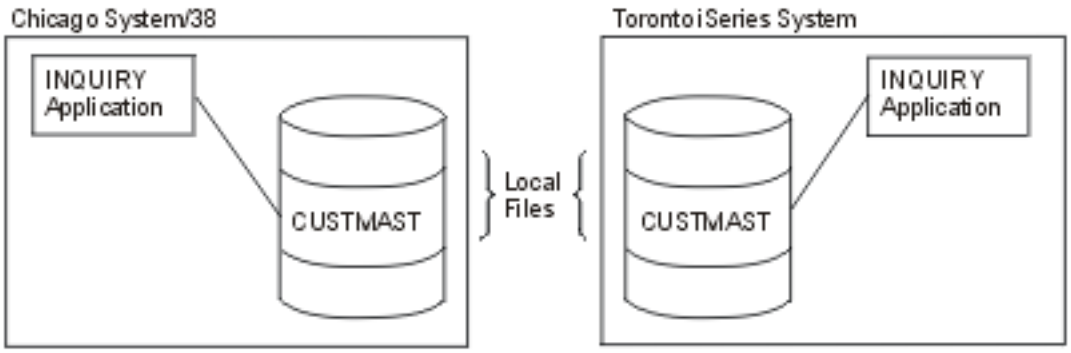


Figure 17. Two non-DDM servers doing local inquiries

The following program (in pseudocode form) is run at each location to access its own primary file named CUSTMAST.

```
       Open CUSTMAST
LOOP:  Prompt for CUSTNO
       If function 1, go to END
       Get customer record
       Display
       Go to LOOP
END:   Close CUSTMAST
       RETURN
```

Using DDM, the CUSTMAST files are consolidated into one file at a centralized location (Philadelphia, in these examples), and then the local files in Chicago and Toronto can be deleted. The inquiry program used at each remote location and at the central location to access that file is identical to the program used previously.

To perform *remote* inquiries without changing the program, each of the remote locations need only create a DDM file and use an override command:

```
CRTDDMF   FILE(INQ)  RMTFILE(CUSTMAST)  RMTLOCNAME(PHILLY)

OVRDBF  FILE(CUSTMAST) TOFILE(INQ)
```

The DDM file points to the Philadelphia server as the target server and to the CUSTMAST file as the remote file. The same values for this command can be used at each remote location if they also have a remote location named PHILLY.

Because CUSTMAST is the file name used in the program, the Override with Database File (OVRDBF) command must be used to override the nonexistent CUSTMAST file with the DDM file INQ. (If the CUSTMAST file still exists on the local server, the override is needed to access the central server's primary file; without it, the local file is accessed.)

The figure below shows the same two servers accessing the centralized CUSTMAST file by using their DDM files, each named INQ.

An alternative to this approach is to leave the CUSTMAST files on the Chicago and Toronto servers and use them for nonessential inquiries, such as name and address, and use the central CUSTMAST file in Philadelphia for any changes. The CUSTMAST files on the Chicago and Toronto servers can be changed periodically to the current level of the primary file on the Philadelphia server.

This alternative method will be used in the next example.

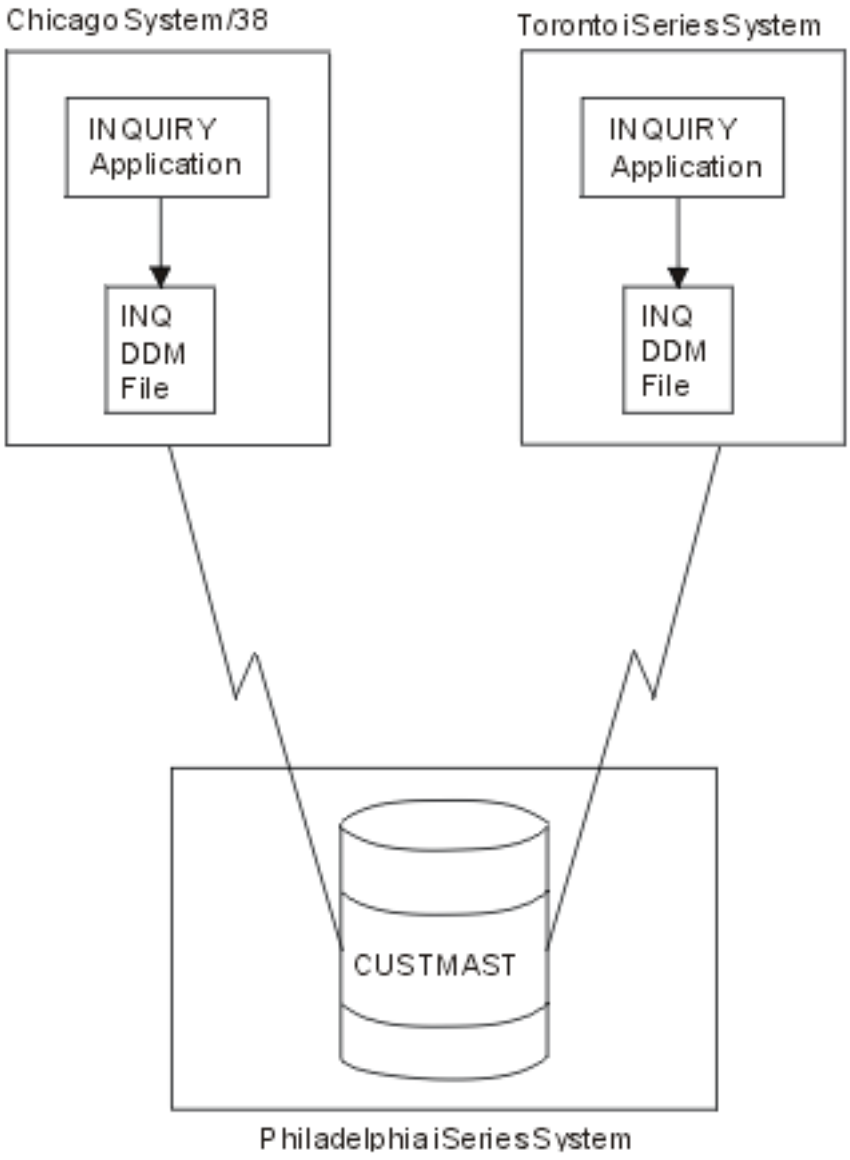


Figure 18. Two DDM servers doing remote inquiries

Related concepts

Control language

DDM example 2: ORDERENT application

This second example shows how multiple locations in a customer's business can process the same order entry application using DDM.

The first task in this example shows how to use DDM to put copies of the same application program on remote servers with one primary file at a central location. The second task in this example shows how to use DDM to copy a file to a remote server.

DDM example 2: Central server ORDERENT files

At the central site of Philadelphia, the four files shown in the figure are being used by the ORDERENT application program.

At the central server, the CUSTMAST file is a physical file that is the primary file of customer data for all locations. The CUSTMST2 file is a logical file that is based on the CUSTMAST physical file. Using a logical file at the central server provides at least two advantages:

- The same program, ORDERENT, can be used *without* change by the central server and by each of the remote servers.
- The data can be accessed through a separate file and cannot keep a customer's primary record locked for the duration of the order.

The four files at the central site are used as follows:

- The CUSTMAST file contains all the data about all its customers. After a customer order is completed, the CUSTMAST file is changed with all the new information provided by that order.
- The CUSTMST2 file, which is a logical file at the central server, is used at the beginning of a customer order. When an operator enters a customer number, the program reads the customer data from the CUSTMST2 logical file, but the data actually comes from the primary file, CUSTMAST.
- The INVEN file contains the current quantities of all items available for sale to customers. When the operator enters an item number and quantity ordered, the corresponding primary item in the INVEN file is changed.
- The DETAIL file is a list of all the individual items ordered; it contains a record for each item and quantity ordered by customers.

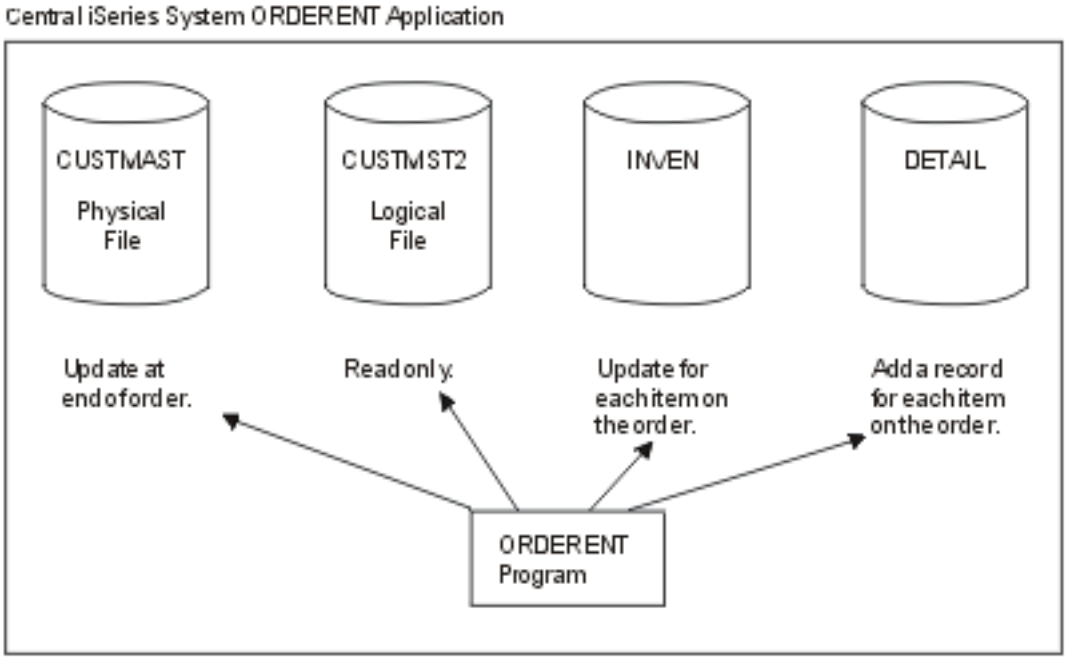


Figure 19. Files used by central server ORDERENT program

DDM example 2: Description of ORDERENT program

Initially, the ORDERENT program exists only in library PGMLIB on the central server (in Philadelphia).

This program does the following items:

- When an order entry operator enters a customer number, ORDERENT reads the customer number, then reads the first member of file CUSTMST2 in the PGMLIB library to find the customer name, address, and other information. The retrieved information is displayed to the operator, and the program asks for an item number and quantity desired.

- When the operator enters an item number and quantity desired and presses the Enter key, the program changes the corresponding primary item in the first member of the INVEN file, and it adds a record to the DETAIL file for each item and quantity entered. The program continues asking for another item number and quantity until the operator ends the program.
- When the operator ends the program, the file CUSTMAST is changed with the information for the entire order. (See the pseudocode of ORDERENT for details.)

For the following examples, it is assumed that all users on the remote servers who need to access CUSTMAST in Philadelphia already have authority to do so, and that those who do not need authority do not have it. In these examples, the iSeries server in Chicago does not have a compiler.

If we want this program to be used at all the remote locations that also stock a physical inventory, the program needs to be sent to each of the remote servers. We can assume that each of the remote servers has its own inventory and primary files INVEN, DETAIL, and CUSTMST2 (which is a copy of CUSTMAST). How the program can be sent to a remote server is described in "DDM example 2: Transfer a program to a target server" on page 153.

Note: By using the code examples, you agree to the terms of the "Code license and disclaimer information" on page 209.

Pseudocode for ORDERENT Program

```
   •
   •
   •
DECLARE CUSTMAST CHANGE
      * Declare file CUSTMAST and allow changing.
DECLARE CUSTMST2 READ
      * Declare file CUSTMST2 as read only.
DECLARE INVEN CHANGE
      * Declare inventory file INVEN and allow changing.
DECLARE DETAIL OUTPUT
      * Declare file DETAIL as output only.
   •
   •
   •
Open CUSTMAST, CUSTMST2, INVEN, and DETAIL files
      * Begin program.
   Show order entry display asking for CUSTNO.
         * Order entry operator enters CUSTNO.
   If function key, go to End.
   Read CUSTNO from display.
      For CUSTNO, return NAME, ADDR, and other
      information from CUSTMST2 file.
   Show NAME, ADDR, and other information on display.
   LOOP: Display 'Item Number ___      Quantity Desired ____'.
         * Order entry operator enters item number and quantity.
      Read ITEMNO and Quantity Desired from display.
      If ITEMNO = 0 then go to LOOPEND.
         Change INVEN with ITEMNO and Quantity Desired.
         Write an item record to the DETAIL file.
      Go to LOOP.
   LOOPEND:  For CUSTNO, change CUSTMAST using
             information in file INVEN.
   End
      * Program has ended.
Close CUSTMAST, CUSTMST2, INVEN, and DETAIL files.
RETURN
```

DDM example 2: Remote server ORDERENT files

The ORDERENT program remains the same at all locations, but the CUSTMST2 file is now a *copy* of the central server's customer primary file CUSTMAST.

By using CUSTMST2 whenever possible for data that does not change often, the amount of communications time needed to process each order entry request can be minimized. The remote ORDERENT program reads the local CUSTMST2 file at the beginning of each order, and then, using DDM, updates the CUSTMAST file on the central server only when an order has been completed.

The other two files, INVEN and DETAIL, have the same functions on each remote server as on the central server.

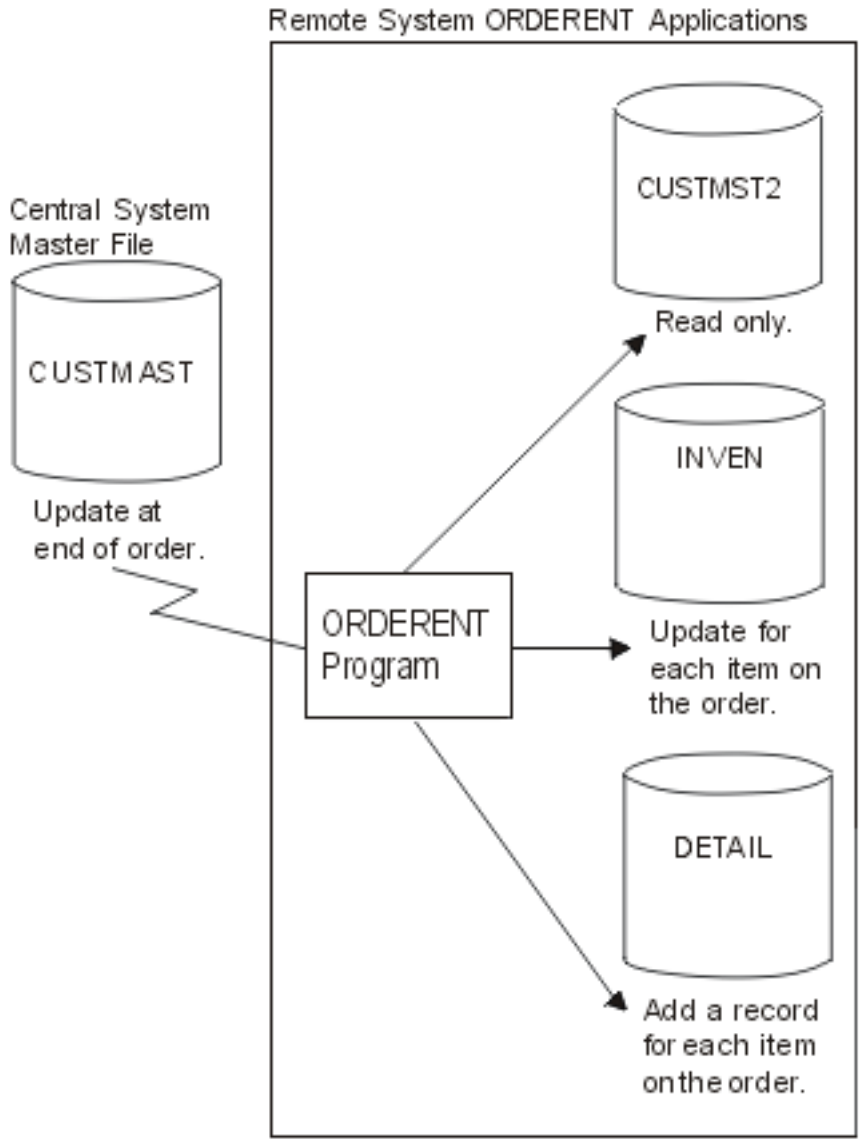


Figure 20. Files used by remote ORDERENT programs

The CUSTMAST file is changed by all locations and contains the most current information for each customer (for constantly changing data such as the customer's account balance). The CUSTMST2 file, which is used for reading data that changes only occasionally (such as name and address), should be changed periodically (once a week, for example), by recopying the CUSTMAST file into it. Task 2 of this example explains one way to do this.

DDM example 2: Transfer a program to a target server

In this task, the central server in the DDM network, located in Philadelphia, sends a program named ORDERENT to a remote System/38 in Chicago.

The program ORDERENT is transferred from the Philadelphia server to the user in Chicago whose user ID is ANDERSON CHICAGO, and then the program is set up so that ORDERENT in Chicago changes the CUSTMAST file in library PGMLIB on the central server in Philadelphia. The read-only function is performed against the local file (in Chicago) and the change is done in the remote file (in Philadelphia).

For this task, two methods are shown for transferring the ORDERENT program in Philadelphia to the remote server in Chicago. Basically, the same sets of commands are used in both methods, except that the second group of commands used in the pass-through method are embedded in Submit Remote Command (SBMRMTCMD) commands used in the SBMRMTCMD command method.

- The first method uses pass-through and object distribution, allowing the operator on the source server to set up both servers without involving the target system operator or using the SBMRMTCMD command. This method can be used only for iSeries servers or System/38s.
- The second method uses the SBMRMTCMD command because, in this task, the target server is a System/38. (The SBMRMTCMD command can be used when the target server is an iSeries server or a System/38.)

DDM example 2: Pass-through method:

One set of commands is entered on the source server, a pass-through session is started with the target server, and a second set of commands is entered on the source server and *run* on the target server.

The following commands are issued on the source server in Philadelphia:

```
CRTSAVF  FILE(TRANSFER)
SAVOBJ   OBJ(ORDERENT)  LIB(PGMLIB)  SAVF(TRANSFER)
   UPDHIST(*NO)  DTACPR(*YES)
SNDNETF  FILE(TRANSFER)  TOUSRID(ANDERSON CHICAGO)
```

Next, a pass-through session is started between the Philadelphia and Chicago servers with the Begin Pass-Through (BGNPASTHR) command. The session is used at the source server to enter the following commands, which are run on the target server:

```
CRTSAVF  FILE(RECEIVE)
RCVNETF  FROMFILE(TRANSFER)  TOFILE(RECEIVE)
CRTLIB   LIB(PGMLIB)
RSTOBJ   OBJ(ORDERENT)  SAVLIB(PGMLIB)  SAVF(RECEIVE)
CRTDDMF  FILE(CUSTMAST.PGMLIB)  RMTFILE(*NONSTD 'PGMLIB/CUSTMAST')
   DEVD(PHILLY)
```

These commands create a save file named RECEIVE, into which the TRANSFER file is copied after it is received as a network file from the source server in Philadelphia. A library is created on the Chicago server and the RECEIVE file is restored as the ORDERENT program in the newly created library named PGMLIB. Lastly, a DDM file is created on the Chicago server which allows the Chicago server to access the CUSTMAST file on the Philadelphia server (remote location named PHILLY).

Related concepts


Remote Work Station Support PDF

DDM example 2: SBMRMTCMD command method:

Commands needed to accomplish the task are entered at the source server. The source server sends commands that are needed on the target iSeries server by using the Submit Remote Command (SBMRMTCMD) command between the servers.

The following commands are issued on the source server in Philadelphia to send the ORDERENT program to the target server in Chicago:

```
CRTSAVF  FILE(TRANSFER)
SAVOBJ   OBJ(ORDERENT)  LIB(PGMLIB)  SAVF(TRANSFER)
   UPDHIST(*NO)
SNDNETF  FILE(TRANSFER)  TOUSRID(ANDERSON CHICAGO)
CRTDDMF  FILE(CHICAGO)  RMTFILE(xxxxx)  RMTLOCNAME(CHIC)
SBMRMTCMD  CMD('CRTSAVF  FILE(RECEIVE)')  DDMFILE(CHICAGO)
SBMRMTCMD  CMD('RCVNETF  FROMFILE(TRANSFER)
   TOFILE(RECEIVE)')  DDMFILE(CHICAGO)
SBMRMTCMD  CMD('CRTLIB  LIB(PGMLIB)')  DDMFILE(CHICAGO)
SBMRMTCMD  CMD('RSTOBJ  OBJ(ORDERENT)  SAVLIB(PGMLIB)
   SAVF(RECEIVE)')  DDMFILE(CHICAGO)
SBMRMTCMD  CMD('CRTDDMF  FILE(CUSTMAST.PGMLIB)
   RMTFILE(*NONSTD "PGMLIB/CUSTMAST")  DEVD(PHILLY)')
   DDMFILE(CHICAGO)
```

These commands create a save file named TRANSFER, which saves the ORDERENT program and then sends it as a network file to the target server in Chicago. There, the commands embedded in the SBMRMTCMD command are used to create a save file (named RECEIVE) on the target server, receive the TRANSFER file, and restore it as ORDERENT into the newly created PGMLIB library. Lastly, a DDM file is created on the Chicago server which allows the Chicago server wants to access the CUSTMAST file on the Philadelphia server. The Create DDM File (CRTDDMF) command is in System/38 syntax.

After either of these two methods is used to send the ORDERENT program to, and to create the DDM file on, the Chicago server, the ORDERENT program on that server can be used to access the CUSTMAST file on the Philadelphia server.

DDM example 2: Copy a file

After performing the first task in Example 2, you decide you want to copy the current level of the CUSTMAST file (in Philadelphia) to the server in Chicago so you can bring the CUSTMST2 file up to date.

This example assumes that the CUSTMST2 file already exists in Chicago.

The following commands can be used to copy the CUSTMAST file from the Philadelphia server to the CUSTMST2 file on the Chicago server. (These commands are issued on the server in Philadelphia.)

```
CRTDDMF  FILE(PHILLY/COPYMAST)  RMTFILE(*NONSTD 'CUSTMST2.CHICAGO')
   RMTLOCNAME(CHIC)
CPYF  FROMFILE(PGMLIB/CUSTMAST)  TOFILE(PHILLY/COPYMAST)
   MBROPT(*REPLACE)
```

Note: One might assume that, as an alternative method, you can create a DDM file on the source server, use the SBMRMTCMD command to submit a Create DDM File (CRTDDMF) command to the target server, and then *attempt* to use the newly created target DDM file with another SBMRMTCMD command to perform the copy function back to the original server. However, that method *will not work*, because *an iSeries server cannot be both a source and target server within the same job*.

DDM example 3: Access multiple iSeries files

Using the same communications environment as in the previous examples, this example is used to ask inventory questions of identically named files on the two remote iSeries servers and the remote System/38.

To do so, a program must be written (shown here in pseudocode) on the central server that can access the files named LIB/MASTER on the servers in Chicago, in Toronto, and in New York. (In this example, the MASTER files are keyed files, and the first member of each of these files is the one used. Also, data description specifications (DDS) for the MASTER files exist on the central server in Philadelphia.)

The program asks the local order entry operator for an item number (ITEMNO), and returns the quantity-on-hand (QOH) information from the files in Chicago, Toronto, and New York.

The following commands are issued on the server in Philadelphia:

```
CRTDDMF  PGMLIB/CHIFILE  RMTFILE(*NONSTD 'MASTER.LIB')
         RMTLOCNAME(CHIC)
CRTDDMF  PGMLIB/TORFILE  RMTFILE(LIB/MASTER)   RMTLOCNAME(TOR)
CRTDDMF  PGMLIB/NYCFILE  RMTFILE(LIB/MASTER)   RMTLOCNAME(NYC)
```

Here is a sample of the pseudocode to accomplish the task:

Note: By using the code examples, you agree to the terms of the "Code license and disclaimer information" on page 209.

```
DECLARE CHIFILE, TORFILE, NYCFILE INPUT
Open CHIFILE, TORFILE and NYCFILE
LOOP:  Show a display asking for ITEMNO
   Read ITEMNO from the display
      Read record from CHIFILE with the key ITEMNO
      Read record from TORFILE with the key ITEMNO
      Read record from NYCFILE with the key ITEMNO
      Write all QOH values to the display
   If not function key, go to LOOP
Close CHIFILE, TORFILE and NYCFILE
END
```

Before the program is compiled, Override with Database File (OVRDBF) commands can be used to override the three files used in the program with local files that contain the external description formats, identical to the remote files being accessed. Doing so significantly reduces the time required for the compile, since the files on the remote server do not have to be accessed then.

After the program has been compiled correctly, the overrides should be deleted so that the program is able to access the remote files.

An alternative to the use of overrides is to keep the file definitions in a different library. The program can be compiled using the file definitions in that library and then run using the real library.

DDM example 4: Access a file on System/36

This topic shows how the pseudocode program for the previous task can be changed so a MASTER file on the System/36 in Dallas can be accessed in the same way as the MASTER files on the iSeries servers and System/38 in Example 3.

Assume that either you have pass-through to the System/36, or that an operator at the System/36 can make changes, if necessary, on the System/36 for you.

The following command is issued on the server in Philadelphia:

```
CRTDDMF  FILE(PGMLIB/DALFILE)  RMTFILE(MASTER)
   RMTLOCNAME(DAL)  ACCMTH(*KEYED)
```

Because the remote file referred to by the DDM file named DALFILE is on a System/36, either of two things must be done:

- The record format of the remote file must be described in the program; that is, it must be a program-described file.
- The program must be compiled with the program referring to a local iSeries file instead of the System/36 file. This local file must have the same record format name as the DDM file name. Note that the local file need not contain any data records.

Here is a sample of the pseudocode to accomplish the task:

Note: By using the code examples, you agree to the terms of the "Code license and disclaimer information" on page 209.

```
DECLARE CHIFILE, TORFILE, NYCFILE, DALFILE INPUT
Open CHIFILE, TORFILE, NYCFILE and DALFILE
LOOP:  Show a display asking for ITEMNO
Read ITEMNO from the display
   Read record from CHIFILE with the key ITEMNO
   Read record from TORFILE with the key ITEMNO
   Read record from NYCFILE with the key ITEMNO
   Read record from DALFILE with the key ITEMNO
   Write all QOH values to the display
   If not function key, go to LOOP
Close CHIFILE, TORFILE, NYCFILE and DALFILE
END
```

Related reference

"Data description specifications considerations for DDM" on page 108
Data description specifications (DDS), which is used to externally describe the fields and record formats, can also be used with DDM to describe the file and record formats of a remote file.

DDM architecture code point attributes

All DDM architecture words are grouped into classes.

Each word in DDM specifies the class to which it belongs with a two-byte hexadecimal code point. The code point is used to reduce the number of bytes needed to identify the class of a word in main storage and in data streams. The code point specifies the location of the class of the word in the *DDM Architecture: Reference*, SC21-9526.

When a system message is displayed, a reference is made to a hexadecimal code point. This appendix provides a list of those code points arranged by hexadecimal value.

Table 10. DDM architecture code points attributes

Code point (Hexadecimal)	Term	Message text
0001	ASSOCIATION	Name with value association
0002	MINLVL	Minimum level
0003	BIN	Binary integer number
0004	BITDR	A single bit data representation
0005	BITSTRDR	Bit string data representation
0006	BOOLEAN	Truth state
0007	QLFATT	Qualified attribute
0008	CHRDR	A graphic character data representation
0009	CHRSTRDR	Character string data representation
000A	CLASS	Object descriptor
000B	CNSVAL	Constant value
000C	CODPNT	Code point attribute
000D	COLLECTION	Collection object
000E	COMMAND	Command
000F	DATE	Date and time
0011	DFTVAL	Default value attribute
0012	DGTSTRDR	Digit string data representation
0013	DGTDR	Numeric character data representation
0014	NOTE	Note attribute
0015	ENULEN	Enumerated length attribute
0016	ENUVAL	Enumerated value attribute
0017	ERROR	Error severity code
0018	FALSE	False state

Table 10. DDM architecture code points attributes (continued)

Code point (Hexadecimal)	Term	Message text
0019	HELP	Help text
001A	HEXDR	Hexadecimal number data representation
001B	HEXSTRDR	Hexadecimal string data representation
001C	IGNORABLE	Ignorable value attribute
001D	INDEX	File index
001E	INFO	Information only severity code
001F	LENGTH	Length of value attribute
0020	LETTER	Alphabetic character
0021	MAXLEN	Maximum length attribute
0022	MAXVAL	Maximum value attribute
0023	MENU	Menu
0024	MAGNITUDE	Linearly comparable scalar
0025	MINLEN	Minimum length attribute
0026	MINVAL	Minimum value attribute
0027	NAME	Name
002A	NIL	Nil object
002B	NUMBER	Number
002C	OBJECT	Architected data entity
002D	OPTIONAL	Optional value attribute
002E	PRMDMG	Permanent damage severity code
0031	REPEATABLE	Repeatable variable attribute
0032	REQUIRED	Required value attribute
0033	RESERVED	Reserved value attribute
0034	SCALAR	Scalar object
0036	SPCVAL	Special value attribute
0037	SPRCLS	Superclass
0038	STRING	String
003A	SEVERE	Severe error severity code
003B	TRUE	True state
003C	DATA	Encoded information
003D	WARNING	Warning severity code
003E	ACCDMG	Access damage severity code
003F	SESDMG	Session damage severity code
0040	ENUCLS	Enumerated class attribute
0041	CMDTRG	Command target
0042	BINDR	Binary data representation
0043	BYTDR	An 8-bit value data representation
0044	BYTSTRDR	Byte string data representation
0045	TITLE	A brief description
0046	ATTLST	Attribute list
0047	DEFLST	Definition list
0048	DEFINITION	Definition
0049	INHERITED	Inherited definitions attribute
004A	STSLST	Term status array
004B	ARRAY	Object array
004C	ORDCOL	Ordered collection
004D	ELMCLS	Element of enumerated class attribute
0050	CONSTANT	Constant value
005D	INSTANCE_OF	Instance of
0064	CODPNTDR	Code point data representation
0065	DATDR	Date and time data
0066	NAMDR	Name date
0067	MTLEXC	Mutually exclusive attribute

Table 10. DDM architecture code points attributes (continued)

Code point (Hexadecimal)	Term	Message text
1001	CLRFIL	Clear file
1002	CLOSE	Close file
1003	CRTAIF	Create alternative index file
1004	CLSDRC	Close directory
1005	FRCBFF	Force buffers
1006	DELFIL	Delete file
1007	GETREC	Get record
1008	INSRECNB	Insert by record number
1009	LSTFAT	List file attributes
100A	GETDRCEN	Get directory entry
100B	LCKFIL	Lock file
100C	SETUPDNB	Set update intent by record number
100D	OPEN	Open file
100E	DELREC	Delete record
100F	MODREC	Modify record
1010	OPNDRC	Open directory
1011	RNMDRC	Rename directory
1013	SETNBR	Set cursor to record number
1014	SETBOF	Set cursor to beginning of file
1015	SETEOF	Set cursor to end of file
1016	SETFRS	Set cursor to first record
1017	SETKEY	Set cursor by key
101B	SETUPDKY	Set update intent by key value
101C	SETLST	Set cursor to last record
101D	SETMNS	Set cursor minus
101E	SETNXT	Set cursor to next record
101F	SETPLS	Set cursor plus
1020	SETPRV	Set cursor to previous record
1023	UNLFIL	Unlock file
1024	INSRECEF	Insert record at end of file
1025	SETKEYLM	Set key limits
1028	CRTDIRF	Create direct file
1029	CRTKEYF	Create keyed file
102A	CRTSEQF	Create sequential file
102C	DCLFIL	Declare file
102D	DELDCL	Delete declared name
102E	LODRECF	Load records into file
1032	INSRECKY	Insert by key value
1036	RNMFIL	Rename file
1037	SETKEYFR	Set cursor to first record in key sequence
1039	SETKEYLS	Set cursor to last record in key sequence
103B	SETKEYNX	Set cursor to next record in key sequence
103C	SETKEYPR	Set cursor to previous record in key sequence
103D	UNLIMPLK	Unlock implicit record lock
1040	ULDRECF	Unload records from file
1041	EXCSAT	Exchange server attributes
1042	SETNXTKE	Set cursor to next record with equal key
1043	CHGFAT	Change file attributes
1044	CRTDRC	Create directory
1045	CRTSTRF	Create stream file
1047	GETSTR	Get stream
1048	LCKSTR	Lock stream

Table 10. DDM architecture code points attributes (continued)

Code point (Hexadecimal)	Term	Message text
1049	PUTSTR	Put stream
104B	UNLSTR	Unlock stream
104C	LODSTRF	Load stream file
104D	ULDSTRF	Unload stream file
104E	CPYFIL	Copy file
104F	CHGCD	Change current directory
1050	CHGEOF	Change end-of-file
1051	DELDRC	Delete directory
1052	QRYSPC	Query space available
1053	SBMSYSCMD	Submit System Command command
1059	QRYCD	Query current directory
1101	BGNNAM	Beginning search name
1102	FILATTRL	File attribute request list
1103	BASFILNM	Base file name
1104	BYPINA	Bypass inactive record
1105	DELDRCOP	Delete directory option
1108	FILCRTDT	File creation date
1109	CSRDSP	Cursor displacement
110A	RELOPR	Relational operator
110B	EOFNBR	End of file record number
110C	FILEXNSZ	File extent size
110D	FILEXPDT	File expiration date
110E	FILNAM	File name
110F	FILSIZ	File size
1110	FILCLS	File class
1111	DFTRECOP	Default record option
1113	LSTACCDT	Last access date
1114	KEYDEF	Key definition
1115	KEYVAL	Key value
1116	MAXGETCN	Maximum get count
1117	FILMAXEX	File maximum number of extents
1118	PRPSHD	Prepare shadow
1119	OVRDTA	Overwrite data
111A	RECCNT	Record count
111B	DELCP	Deletion capability
111C	RECLEN	Record length
111D	RECNBR	Record number
111E	RECNBRFB	Record number feedback
1122	SHDEXS	Shadow exists
1123	SHDONL	Shadow only
1124	UPDCSR	Update cursor
1125	SHDPRC	Shadow processing
1126	ERRFILNM	Error file name
1128	RTNREC	Return record
1129	STRORD	Stream order
112A	FILPRT	File protected
112B	EOFOFF	End of file offset
112F	KEYHLM	Key high limit
1130	KEYLLM	Key low limit
1132	FILHDD	Hidden file
1133	FILSYS	System file
1134	ACCINTLS	Access intent list
1136	DCLNAM	Declared name

Table 10. DDM architecture code points attributes (continued)

Code point (Hexadecimal)	Term	Message text
1137	DUPFILOP	Duplicate file option
1139	FILBYTCN	File byte count
113A	FILCHGDT	File change date
113B	FILEXNCN	File extent count
113C	FILINISZ	Initial file size
113D	KEYDUPCP	Duplicate keys capability
113F	PRCCNVCD	Conversational protocol error code
1142	RECLENCL	Record length class
1143	RLSFILLK	Release file lock
1145	RQSFILLK	Requested file lock
1146	UPDINT	Update intent
1147	SRVCLSNM	Server class name
1148	RTNCLS	File retention class
1149	SVRCOD	Severity code
114A	SYNERRCD	Syntax error code
114B	TEXT	Text character string
114C	WAIT	Wait for lock
114D	FILSHR	File sharing
114E	ACCMTHCL	Access method class
114F	NEWFILNM	New file name
1150	BYPDMG	Bypass damaged records
1151	LCKMGRNM	Lock manager name
1152	AGNNAM	Agent name
1153	SRVDGN	Server diagnostic information
1154	ALCINIEX	Allocate initial extent
1155	RTNINA	Return inactive record
1156	ALWINA	Allow cursor to be set to inactive record
1157	MAXOPN	Maximum number of files opened
1159	MAXARNB	Maximum active record number
115A	SRVRLSLV	Server product release level
115B	CSRPOSST	Cursor position status
115C	DTALCKST	Data lock status
115D	SPVNAM	Supervisor name
115E	EXTNAM	External name
115F	HLDCSR	Hold cursor position
1160	KEYVALFB	Key value feedback
1161	ALWMODKY	Allow modified keys
1162	ACCORD	Access order
1163	RLSUPD	Release update intent
1164	KEYDEFCD	Key definition error code
1165	DRCNAM	Directory name
1166	MODCP	File modify capability
1169	STRLEN	Stream length
116A	STRPOS	Position of a stream in a stream file
116B	STRSIZ	Stream file size
116D	SRVNAM	Server name
1174	SPCUNT	Space units
1175	SPCTTL	Total space
117E	SPCAVL	Available space
1183	STROFF	Stream offset
118A	LSTARCDT	Last archived date
118B	RQSSTRLK	Request stream lock
118C	STRLOC	Substream location

Table 10. DDM architecture code points attributes (continued)

Code point (Hexadecimal)	Term	Message text
118D	CPYNEW	Copy to new file option
118E	CPYOLD	Copy to existing file option
118F	NEWDRCNM	New directory name
1191	GETCP	File get capability
1192	INSCP	File insert capability
1194	FILCHGFL	File change flag
11B8	SYSCMD	System command
11BC	SYSCMDMSG	System command message
11D8	SYCMMGNM	System command manager name
1201	KEYUDIRM	Key update not allowed by different index reply message
1203	SYSCMDRM	System command reply message
1204	DFTRECRM	Default record error
1205	CSRNSARM	Cursor not selecting a record position reply message
1206	DTARECRM	Data record reply message not valid
1207	DUPFILRM	Duplicate file name reply message
1208	DUPKDIRM	Duplicate key different index reply message
1209	DUPKSIRM	Duplicate key same index reply message
120A	DUPRNBRM	Duplicate record number reply message
120B	ENDFILRM	End of file reply message
120C	FILFULRM	File is full reply message
120D	FILIUSRM	File in use reply message
120E	FILNFNRM	File not found reply message
120F	FILSNARM	File space not available reply message
1210	MGRLVLRM	Manager level conflict reply message
1211	FILNOPRM	File not opened reply message
1212	FILNAMRM	File name reply message not valid
1214	SHDEXSRM	Shadow exists reply message
1215	RECLENRM	Record length mismatch reply message
1218	MGRDEPRM	Manager dependency error reply message
121C	CMDATHRM	Not authorized to command reply message
121E	FILTNARM	File temporarily not available reply message
1220	DCLCNFRM	Declare conflict reply message
1221	DRCTNARM	Directory temporarily not available reply message
1224	RECNBRRM	Record number out of bounds reply message
1225	RECNFNRM	Record not found reply message
122D	KEYLENRM	Key length reply message not valid
1230	ACCATHRM	Not authorized to access method reply message
1231	ACCMTHRM	Access method reply message not valid
1232	AGNPRMRM	Permanent agent error reply message
1233	RSCLMTRM	Resource limits reached reply message
1234	BASNAMRM	Base file name reply message not valid
1237	DRCATHRM	Not authorized to directory reply message
123A	EXSCNDRM	Existing condition reply message
123B	FILATHRM	Not authorized to file reply message
123C	INVRQSRM	Invalid request reply message

Table 10. DDM architecture code points attributes (continued)

Code point (Hexadecimal)	Term	Message text
123D	KEYDEFRM	Key definition reply message not valid
123F	KEYUSIRM	Key update not allowed by same index reply message
1240	KEYVALRM	Key value reply message not valid
1242	OPNCNFRM	Open conflict error reply message
1243	OPNEXCRM	Open exclusive by same user reply message
1244	OPNMAXRM	Opens at the same time exceed maximum reply message
1245	PRCCNVRM	Conversational protocol error reply message
1249	RECDMGRM	Record damaged reply message
124A	RECIUSRM	Record in use reply message
124B	CMDCMPRM	Command processing completed reply message
124C	SYNTAXRM	Data stream syntax error reply message
124D	UPDCSRRM	Update cursor error reply message
124E	UPDINTRM	No update intent on record reply message
124F	NEWNAMRM	New file name reply message not valid
1250	CMDNSPRM	Command not supported reply message
1251	PRMNSPRM	Parameter not supported reply message
1252	VALNSPRM	Parameter value not supported reply message
1253	OBJNSPRM	Object not supported reply message
1254	CMDCHKRM	Command check reply message
1255	DUPDCLRM	Duplicate declared name reply message
1256	DCLNAMRM	Declared name reply message not valid
1257	DCLNFNRM	Declared name not found reply message
1258	DRCFULRM	Directory full reply message
1259	RECINARM	Record inactive reply message
125A	FILDMGRM	File damaged reply message
125B	LODRECRM	Load records count mismatch reply message
125C	INTATHRM	Not authorized to open intent for named file reply message
125E	CLSDMGRM	File closed with damage reply message
125F	TRGNSPRM	Target not supported reply message
1260	KEYMODRM	Key value modified after cursor was last set reply message
1261	CHGFATRM	Change file attributes rejected reply message
1262	DRCNAMRM	Directory name not valid
1263	DRCNFNRM	Directory not found reply message
1264	STRIUSRM	Stream in use error
1265	SUBSTRRM	Substream reply message not valid
1266	ACCINTRM	Access intent not valid for access method
1267	DRCIUSRM	Directory in use reply message
1268	STRDMGRM	Stream damaged reply message
1269	DRCENTRM	Directory entry reply message not valid
126A	DUPDRCRM	Duplicate directory name
126B	DRCSNARM	Directory space not available
126C	DTAMAPRM	Data mapping error reply message

Table 10. DDM architecture code points attributes (continued)

Code point (Hexadecimal)	Term	Message text
126E	LODSTRRM	Load stream count mismatch reply message
126F	RECNAVRM	Record not available reply message
1270	DRCNEMRM	Directory not empty reply message
127E	DRCDMGRM	Directory damaged reply message
1282	DRCSUBRM	Directory contains subdirectory reply message
1283	NEWDRNRM	New directory name reply message not valid
1401	ACCMTH	Access method
1402	ACCMTHLS	Access method list
1403	AGENT	Agent
1404	MGRLVLLS	Manager level list
1405	CMBACCAM	Combined access method
1406	CMBKEYAM	Combined keyed access method
1407	CMBRNBAM	Combined record number access method
1408	CMNMGR	Communications manager
140A	RECCSR	Record cursor
140B	DELAI	Delete access intent
140C	DIRFIL	Direct file
140D	DSSFMT	Data stream structure format
140F	KEYFLDDF	Key field definition
1410	EXTENT	File extent
1411	RECFIL	Record file manager
1413	GETGETLK	Get intent willing to share with get intents at the same time
1414	GETMODLK	Get intent willing to share with modify intents at the same time
1415	GETNONLK	Get intent not willing to share with any users at the same time
1416	GETAI	Get access intent
1417	INSAI	Insert access intent
1418	DCAL3P	Document content architecture level three
1419	DRCAM	Directory access method
141A	DRCCSR	Directory cursor
141B	DRCEMP	Directory empty option
141C	DRPSHD	Drop shadow
141E	KEYFIL	Keyed file
1420	SEQASC	Ascending key sequence
1421	SEQDSC	Descending key sequence
1422	LCKMGR	Lock manager
1423	ALTINDF	Alternative index file
1424	FILAL	File attribute list
1425	MODGETLK	Modify intent willing to share with get intents at the same time
1426	MODMODLK	Modify intent willing to share with modify intents at the same time
1427	MODNONLK	Modify intent not willing to share with any users at the same time
1428	MODAI	Modify access intent
1429	OBJDSS	Object data stream structure
142A	PRMFIL	Permanent file
142B	DFTREC	Default record
142C	PCEXE	PC EXE formatted stream file

Table 10. DDM architecture code points attributes (continued)

Code point (Hexadecimal)	Term	Message text
142D	RECINA	Inactive record
142E	RECFIX	Fixed length record
142F	RECIVL	Initially varying length record
1430	RECAL	Record attribute list
1431	RECVAR	Varying length record
1432	RELKEYAM	Relative by key access method
1433	RELRNBAM	Relative by record number access method
1434	RNDKEYAM	Random by key access method
1435	RNDRNBAM	Random by record number access method
1436	RPYDSS	DDM reply data stream structure
1437	RPYMSG	Reply message
1438	RQSCRR	Request correlation identifier
1439	RQSDSS	Request data stream structure
143A	BOF	Beginning of file
143B	SEQFIL	Sequential file
143C	SUPERVISOR	Supervisor
143D	SHRRECLK	Share record lock
143E	TMPFIL	Temporary file
143F	EXCRECLK	Exclusive record lock
1440	SECMGR	Security manager
1441	EOF	End of file
1442	MGRLVL	Manager level
1443	EXCSATRD	Server attributes reply data
1444	CMNAPPC	APPC conversational communications manager
1445	KEYAE	Key after or equal to relational operator
1446	KEYAF	Key after operator
1447	KEYEQ	Key equal relational operator
1448	SERVER	Server
1449	DFTSRCIN	Default source initialization
144A	RECORD	Record
144B	KEYBE	Key before or equal to relational operator
144C	KEYBF	Key before operator
144D	FILIND	File index
144E	ALTINDLS	Alternative index list
144F	FILINDEN	File index entry
1450	DCTIND	Dictionary index
1451	DCTINDEN	Dictionary index entry
1452	MGRNAM	Manager name
1453	MGRADR	Manager address
1454	DRCIND	Directory index
1455	DRCINDEN	Directory index entry
1456	MANAGER	Resource manager
1457	DIRECTORY	Directory file
1458	DICTIONARY	Dictionary
1459	DUPFILDO	Duplicate file reply message duplicate option
145A	EXSCNDDO	Existing condition reply message duplicate option
145C	CLRFILDO	Clear file duplicate option
145D	KEYORD	Key order processing
145E	RNBORD	Record number order processing

Table 10. DDM architecture code points attributes (continued)

Code point (Hexadecimal)	Term	Message text
145F	DFTTRGIN	Default target initialization
1460	DFTINAIN	Default inactive record initialization
1461	DCAFFT	Document content architecture final form text
1462	CPYNCR	Copy with no create option
1463	STRAM	Stream access method
1464	STREAM	Stream
1465	STRFIL	Stream file
1466	CPYDTA	Copy with data option
1467	CPYNDT	Copy with no data option
1468	CURSOR	Access method cursor
1469	STRCSR	Stream cursor
146A	FILE	File manager
1471	DCARFT	Document content architecture revisable form text
1473	MGRLVLN	Manager level number attribute
1479	QRYSPCRD	Query space reply data
147F	SYSCMDMGR	System command manager
1482	CPYAPP	Copy append option
1483	CPYERR	Copy duplicate file error option
1484	CPYRPL	Copy replace option
1485	EXCSTRLK	Exclusive stream lock
1486	SHRSTRLK	Share stream lock
1487	MODSTRLK	Modify stream lock
1488	DRCALL	Delete all files in directory option
1489	DRCANY	Delete any accessible files in directory

DDM commands and parameters

This topic classifies DDM commands and parameters.

For additional information about DDM subsets, see the *DDM Architecture: Implementation Planner's Guide* or the *DDM Architecture: Reference*.

Note: The abbreviation *KB* appears throughout the tables in these topics. It represents a quantity of storage equal to 1024 bytes.

Related reference

"DDM architecture-related restrictions" on page 45
The items listed in this topic are DDM architecture-related restrictions. Therefore, application programs that use these items might have to be changed and recompiled before they can access remote files.

"Variable-length records" on page 118
If your iSeries source server is running OS/400 Version 2 Release 1 Modification 1, DDM supports variable-length record files as defined in the DDM architecture.

Subsets of DDM architecture supported by i5/OS DDM

The iSeries server supports these subsets of the DDM architecture.

Supported DDM file models

iSeries DDM supports these DDM file models.

- Alternate index file (ALTINDF)

- Direct file (DIRFIL)
- Directory file (DIRECTORY)
- Keyed file (KEYFIL)
- Sequential file (SEQFIL)
- Stream file (STRFIL)

By using the above file models, the iSeries server supports access to the iSeries physical and logical files. The following table shows how DDM file models and iSeries data files correspond.

Table 11. iSeries data files

DDM file model	Corresponding iSeries data file
Alternate index file (ALTINDF)	Logical file with one format
Direct file (DIRFIL)	Nonkeyed physical file
Directory file (DIRECTORY)	Folder management services (FMS) folders or data management libraries
Keyed file (KEYFIL)	Keyed physical file
Sequential file (SEQFIL)	Nonkeyed physical file
Stream file (STRFIL)	Folder management services (FMS) document

Alternate Index File (ALTINDF):

i5/OS DDM supports access to a logical file by using the DDM alternate index file model.

A logical file allows access to the data records stored in a physical file by using an alternate index defined over the physical file. Only single format logical files can be accessed through i5/OS DDM. Logical files with select/omit logic can be accessed, but records that are inserted might not be retrievable if they are omitted by the select/omit logic.

Supported record classes

An iSeries alternate index file can have fixed-length record (RECFIX) or variable-length record (RECVAR) for storage.

Once a non-iSeries source server opens a file on the iSeries target using variable-length record access, the iSeries target continues to send and receive variable-length records on all subsequent I/O operations.

Note: i5/OS DDM supports the DDM file transfer commands Load Record File (LODRECFIL) and Unload Record File (ULDRECFIL) for all of the file models except alternate index file.

Direct file (DIRFIL):

i5/OS DDM supports access to nonkeyed physical files via the DDM direct file model.

The support has the following characteristics:

- Delete Capabilities: An iSeries direct file is delete capable or nondelete capable. A nondelete capable file must have an active default record.
- Supported Record Classes: An iSeries direct file can have a fixed-length record (RECFIX) or variable-length record (RECVAR) for storage. Once a non-iSeries source server opens a file on the iSeries target using variable-length record access, the iSeries target continues to send and receive variable-length records on all subsequent I/O operations.

Note: The iSeries server does not support the concept of a direct file. i5/OS DDM creates a direct file by creating a nonkeyed physical file and initializing it, with deleted or active default records, to the maximum size requested. No extensions to the file are allowed.

Directory file (DIRECTORY):

i5/OS DDM supports access to a folder management services folder or a data management library via the DDM directory file model. Folders can be created, opened, renamed, closed, or deleted. Libraries can be created, renamed, or deleted.

Keyed file (KEYFIL):

i5/OS DDM supports access to keyed physical files via the DDM keyed file model.

An iSeries keyed file can have fixed-length record (RECFIX) or variable-length record (RECVAR) for storage. Once a non-iSeries source server opens a file on the iSeries target using variable-length record access, the iSeries target continues to send and receive variable-length records on all subsequent I/O operations.

Sequential file (SEQFIL):

The iSeries server supports access to nonkeyed physical files via the DDM sequential file model.

The support has the following characteristics:

- The sequential file can be delete or nondelete capable on an iSeries server.
- The sequential file on an iSeries server can have a fixed-length record (RECFIX) or variable-length record (RECVAR) for storage. Once a non-iSeries source server opens a file on the iSeries target using variable-length record access, the iSeries target continues to send and receive variable-length records on all subsequent I/O operations.

Stream file (STRFIL):

i5/OS DDM supports access to a folder management services document by means of the DDM stream file model.

Supported DDM access methods

i5/OS DDM supports these DDM access methods. DDM abbreviations for the access methods are given in parentheses.

- Combined access method (CMBACCAM)
- Combined keyed access method (CMBKEYAM)
- Combined record number access method (CMBRNBAM)
- Directory access method (DRCAM)
- Random by key access method (RNDKEYAM)
- Random by record number access method (RNDRNBAM)
- Relative by key access method (RELKEYAM)
- Relative by record number access method (RELRNBAM)
- Stream access method (STRAM)

See the following table for a summary of the access methods that i5/OS DDM supports for each DDM file model. For a description of these access methods, refer to the *DDM Architecture: Implementation Planner's Guide, GC21-9528*.

Table 12. Supported access methods for each DDM file model

Term	Access method	DDM file models					
		Sequential file	**Direct file**	**Keyed file**	**Alternate index file**	**Stream file**	**Directory file**
CMBACCAM	Combined access	N	T	T	N		
CMBKEYAM	Combined keyed			T	T		
CMBRNBAM	Combined record number	T	T	T	N		
DRCAM	Directory						T
RELKEYAM	Relative by key			T	T		
RELRNBAM	Relative by record number	T	T	T	N		
RNDKEYAM	Random by key			T	T		
RNDRNBAM	Random by record number	T	T	T	N		
STRAM	Stream					T	
Note: **N** = Not supported **T** = Target DDM supported **Blank** = Not applicable							

DDM commands and objects

These topics describe the DDM command parameters that an iSeries server supports for each DDM architecture command.

For more detailed information about these parameters, see the *DDM Architecture: Reference*, SC21-9526.

The description of the commands might include:

- Limitations for the use of each command
- Objects that the source server might send to the target server
- Objects that the target server might return to the source server
- DDM parameters that the iSeries server supports for the command and how the iSeries server responds to each parameter

The commands listed here are supported. Level 1.0, Level 2.0 and Level 3.0 indicate which level of the DDM architecture is supported by the commands.

CHGCD (Change Current Directory) Level 2.0

This command changes the current path. The path is a string of folders. The current path is added to the front of a file or directory name if it does not start with a slash.

This command is not sent by a source iSeries server.

Parameter name	Source	Target
AGNNAM	N/A	Ignored
DRCNAM[1]	N/A	iSeries name

Parameter name	Source	Target

[1] Name formats are server defined. The architecture specifies that a directory name length of zero indicates the root directory for the Change Current Directory command. For other commands, a directory name length of zero indicates the *current* directory, which might or might not be the root directory at the time the command is issued.

CHGEOF (Change End of File) Level 2.0 and Level 3.0

This command changes the end-of-file mark of a document. The end may be truncated or expanded.

A source iSeries server does not send this command.

Parameter name	Source	Target
DCLNAM	N/A	Program defined
EOFNBR	N/A	Supported
EOFOFF	N/A	Supported

CHGFAT (Change File Attribute) Level 2.0

This command changes the attributes of a file, document, or folder.

Parameter name	Stream file	Directory	Sequential, direct, and keyed files	Alternate index file
DTAFMT	T			
FILCHGDT	T	T	N	N
FILCHGFL	T	N	N	
FILINISZ	N		S, T	
FILEXNSZ	N		S, T	
FILEXPDT			S, T	
FILHDD	T	T	N	N
FILMAXEX	N		S, T	
FILPRT	T	N		
FILSYS	T	T	N	N
DELCP			N	N
GETCP	T		N	
INSCP			N	
MODCP	T		N	
TITLE	T	T	S, T	S, T

N = Not supported

T = Target DDM supported

S = Source DDM supported

Blank = Not applicable

CLOSE (Close File) Level 1.0 and Level 2.0

This command ends the logical connection between the source server and the data set accessed on the target server. Once the target DDM begins running this command, it must close the data set regardless of the reply message returned.

Parameter name	Source	Target
DCLNAM	Program defined	Program defined
SHDPRC	Not sent	Supported

Note: Names are implementation defined.

CLRFIL (Clear File) Level 1.0 and Level 2.0

This command clears an existing file and re-initializes it as if it had just been created.

Parameter name	Source	Target
FILNAM	Target defined	iSeries server
OVRDTA	Not sent	False only

Name formats are server defined.

CLSDRC (Close Directory) Level 2.0

This command closes a folder. This command is not sent by a source iSeries server.

Parameter name	Source	Target
DCLNAM	N/A	Program defined

Names are implementation defined.

CPYFIL (Copy File) Level 2.0

This command copies one document to another document. If the new document does not exist, it might be created. This command is not sent by a source iSeries server.

Parameter name	Source	Target
ACCORD	N/A	Ignored
BYPDMG	N/A	Ignored
BYPINA	N/A	Ignored
CPYNEW[1]	N/A	Supported
CPYOLD[2]	N/A	Supported
DCLNAM[3]	N/A	Program defined
FILNAM[4]	N/A	iSeries name
NEWFILNM[4]	N/A	iSeries name

1. CPYNDT only supported parameter value. All others are rejected with VALNSPRM.
2. CPYERR only supported parameter value. All others are rejected with VALNSPRM.
3. Names are implementation defined.
4. Name formats are server defined.

CRTAIF (Create Alternate Index File) Level 1.0 and Level 2.0

This command creates an alternate index file on the target server.

Parameter name	Source	Target
BASFILNM[1]	Program defined	iSeries name
DUPFILOP	Not sent	Supported
FILCLS[2]	Not sent	Ignored
FILHDD	Not sent	Ignored
FILNAM[3]	Program defined	iSeries name
FILSYS	Not sent	Ignored
KEYDEF[4]	Sent	Supported
KEYDUPCP	Sent	Supported

Parameter name	Source	Target
RTNCLS[5]	Not sent	Supported
TITLE	Sent	Supported

[1] Name formats are server defined.

[2] Only ALTINDF is valid for CRTAIF command.

[3] Name formats are server defined.

[4] iSeries maximum key length is 2000.

[5] Library QTEMP is used for temporaries.

CRTDIRF (Create Direct File) Level 1.0 and Level 2.0

This command creates a direct file on the target server.

Parameter name	Source	Target
ALCINIEX	Sent	Ignored
DCLNAM[1]	Not sent	Supported
DELCP[2]	Sent	Supported
DFTREC	Sent	Supported
DFTRECOP	Sent	Supported
DUPFILOP	Not sent	Supported
FILCLS[3]	Not sent	Ignored
FILEXNSZ[4]	Sent	Supported
FILEXPDT[5]	Sent	Supported
FILHDD	Not sent	Ignored
FILINISZ[4]	Sent	Supported
FILMAXEX[6]	Sent	Supported
FILNAM[7]	Program defined	iSeries name
FILSYS	Not sent	Ignored
GETCP	Sent	Supported
INSCP[8]	Sent	Supported
MODCP	Sent	Supported
RECLEN[9]	Sent	Supported
RECLENCL	Sent	Supported
:row.	RTNCLS[10]	Not sent
Supported		
TITLE	Sent	Supported

[1] Names are implementation defined.

[2] Value must be TRUE unless DFTRECOP (DFTSRCIN) is specified.

[3] Only DIRFIL is valid for CRTDIRF command.

[4] iSeries default is 1,000 records.

[5] iSeries default is *NONE.

[6] iSeries default is 3.

[7] Name formats are server defined.

[8] Only TRUE is valid.

[9] iSeries maximum record length = 2**15-2. (2 to the power of 15 minus 2)

[10] Library QTEMP is used for temporaries.

CRTDRC (Create Directory) Level 2.0

This command creates folders or libraries on the target server, based on the name received. This command is not sent by a source iSeries server.

Parameter name	Source	Target
DCLNAM[1]	N/A	Program defined
DRCNAM[2]	N/A	iSeries name
FILCLS[3]	N/A	Ignored
FILPRT[4]	N/A	Supported
RTNCLS	N/A	PRMFIL only
TITLE	N/A	Supported

[1] Names are implementation defined.

[2] Name formats are server defined.

[3] Only DIRECTORY is valid for CRTDRC command.

[4] FALSE only for libraries.

CRTKEYF (Create Keyed File) Level 1.0 and Level 2.0

This command creates a keyed file on the target server.

Parameter name	Source	Target
ALCINIEX	Sent	Ignored
DCLNAM[1]	Not used	Supported
DELCP	Sent	Supported
DFTREC	Not sent	Supported
DFTRECOP	Not sent	Supported
DUPFILOP	Not sent	Supported
FILCLS[2]	Not sent	Ignored
FILEXNSZ[3]	Sent	Supported
FILEXPDT[4]	Sent	Supported
FILHDD	Not sent	Ignored
FILINISZ[3]	Sent	Supported
FILMAXEX[5]	Sent	Supported
FILNAM[6]	Program defined	iSeries name
FILSYS	Not sent	Ignored
GETCP	Sent	Supported
INSCP	Sent	Supported
KEYDEF[7]	Sent	Supported
KEYDUPCP	Sent	Supported
MODCP	Sent	Supported
RECLEN[8]	Sent	Supported
RECLENCL	Sent	Supported
RTNCLS[9]	Not sent	Supported
TITLE	Sent	Supported

Parameter name	Source	Target
[1]	Names are implementation defined.	
[2]	Only KEYFIL is valid for CRTKEYF command.	
[3]	iSeries default is 1,000 records.	
[4]	iSeries default is *NONE.	
[5]	iSeries default is 3.	
[6]	Name formats are server defined.	
[7]	iSeries maximum key length is 2000.	
[8]	iSeries maximum record length = 2**15-2. (2 to the power of 15 minus 2)	
[9]	Library QTEMP is used for temporaries.	

Note: When a CRTKEYF request is received by an iSeries target server, the new keyed file reuses deleted records when it is created. If duplicate keys are allowed (KEYDUPCP=TRUE sent), the order of the duplicate keys is not guaranteed.

CRTSEQF (Create Sequential File) Level 1.0 and Level 2.0

This command creates a sequential file on the target server.

Parameter name	Source	Target
ALCINIEX	Sent	Ignored
DCLNAM[1]	Not sent	Supported
DELCP	Sent	Supported
DFTREC	Not sent	Supported
DFTRECOP	Not sent	Supported
DUPFILOP	Not sent	Supported
FILCLS[2]	Not sent	Ignored
FILEXNSZ[3]	Sent	Supported
FILEXPDT[4]	Sent	Supported
FILHDD	Not sent	Ignored
FILINISZ[3]	Sent	Supported
FILMAXEX[5]	Sent	Supported
FILNAM[6]	Program defined	iSeries name
FILSYS	Not sent	Ignored
GETCP	Sent	Supported
INSCP	Sent	Supported
MODCP	Sent	Supported
RECLEN[7]	Sent	Supported
RECLENCL	Sent	Supported
RTNCLS[8]	Not sent	Supported
TITLE	Sent	Supported

Parameter name	Source	Target

[1] Names are implementation defined.

[2] Only SEQFIL is valid for CRTSEQF command.

[3] iSeries default is 1,000 records.

[4] iSeries default is *NONE.

[5] iSeries default is 3.

[6] Name formats are server defined.

[7] iSeries maximum record length = 2**15-2. (2 to the power of 15 minus 2.)

[8] Library QTEMP is used for temporaries.

CRTSTRF (Create Stream File) Level 2.0

This command creates a stream file on the target server. This command is not sent by a source iSeries server.

Parameter name	Source	Target
ALCINIEX	N/A	Ignored
DCLNAM[1]	N/A	Program defined
DTAFMT	N/A	Supported
DUPFILOP	N/A	Supported
FILCLS[2]	N/A	Ignored
FILEXNSZ	N/A	Ignored
FILEXPDT	N/A	Ignored
FILHDD	N/A	Supported
FILINISZ	N/A	Ignored
FILMAXEX	N/A	Ignored
FILNAM[3]	N/A	iSeries name
FILPRT	N/A	Supported
FILSYS	N/A	Supported
GETCP	N/A	Supported
MODCP	N/A	Supported
RTNCLS	N/A	Supported
TITLE	N/A	Supported

[1] Names are implementation defined.

[2] Only STRFIL is valid for CRTSTRF command.

[3] Name formats are server defined.

DCLFIL (Declare File) Level 1.0 and Level 2.0

This command associates a declared name (DCLNAM) with a collection of object-oriented parameters in the target agent. This collection is stored by the receiving agent for later use. At the time it is received, the command does not affect objects currently opened by the agent. The primary access to the DCLFIL collection is the DCLNAM parameter.

Parameter name	Source	Target
AGNNAM[1]	Not sent	Ignored
DCLNAM[2]	Program defined	Program defined
DRCNAM[3]	Not sent	iSeries name
FILEXNSZ[4]	Not sent	Ignored
FILMAXEX[4]	Not sent	Ignored

Parameter name	Source	Target
FILNAM[3]	Program defined	iSeries name

[1] Only one agent on an iSeries server.

[2] Names are implementation defined.

[3] Name formats are server defined.

[4] Create value is used.

DELDCL (Delete Declared Name) Level 1.0

This command deletes a declared agent name.

Parameter name	Source	Target
AGNNAM	Not sent	Ignored
DCLNAM[1]	Program defined	Program defined

[1]Names are implementation defined.

DELDRC (Delete Directory) Level 2.0

This command deletes a folder or a library. This command is not sent by a source iSeries server.

Parameter name	Source	Target
DELDRCOP[1]	N/A	DRCEMP or DRCANY
DRCNAM[2]	N/A	iSeries name
OVRDTA	N/A	FALSE only

[1] DRCALL not supported.

[2] Name formats are server defined. Generic names are not supported.

DELFIL (Delete File) Level 1.0 and Level 2.0

This command deletes a file or document.

Parameter name	Source	Target
FILNAM[1]	Target defined generics allowed	iSeries name
OVRDTA[2]	Not sent	FALSE only
SHDONL[3]	Not sent	Supported

[1] Name formats are server defined. Generic names are only allowed for documents.

[2] The iSeries server does not support overwriting.

[3] FALSE only for files.

DELREC (Delete Record) Level 1.0

This command deletes the record that currently has an update intent placed on it. It does this without affecting the current cursor position.

Parameter name	Source	Target
DCLNAM[1]	Program defined	Program defined

[1]Names are implementation defined.

EXCSAT (Exchange Server Attributes) Level 1.0 and Level 2.0

This command exchanges information between servers, such as the server's class name, architectural level of each class of managers it supports, server's product release level, server's external name, and server's name.

Parameter name	Source	Target
EXTNAM	Sent	Supported
MGRLVLS	Sent	Supported
SPVNAM	Not sent	Ignored
SRVCLSNM	Sent	Supported
SRVNAM	Sent	Supported
SRVRLSLV	Sent	Supported

The following reply object is returned:

EXCSATRD
Server attributes reply data

FILAL and FILATTRL (File Attribute List) Level 1.0, Level 2.0, and Level 3.0

This is a list of file attributes that DDM might request on an LSTFAT, OPEN, or GETDRCEN. Some parameters are only valid for specific file types.

Table 13. File attribute list

Parameter name	Source	Target
ACCMTHLS	Requested	Supported
BASFILNM[1]	Requested	iSeries name
DELCP	Requested	Supported
DFTREC	Requested	Supported
DTAFMT	Not requested	Supported
EOFNBR	Requested	Supported
EOFOFF	Not requested	Supported
FILBYTCN	Not requested	Supported
FILCHGDT	Requested	Supported
FILCHGFL	Not requested	Supported
FILCLS	Requested	Supported
FILCRTDT	Requested	Supported
FILEXNCN	Requested	Supported
FILEXNSZ	Requested	Supported
FILEXPDT	Requested	Supported
FILHDD	Not requested	Supported
FILINISZ	Requested	Supported
FILMAXEX	Requested	Supported
FILNAM	Requested	Supported
FILPRT	Not requested	Supported
FILSIZ	Requested	Supported
FILSYS	Not requested	Supported
GETCP	Requested	Supported
INSCP	Requested	Supported
KEYDEF	Requested	Supported
KEYDUPCP	Requested	Supported
LSTACCDT	Not requested	Not supported
LSTARCDT	Requested	Supported
MAXARNB	Requested	Not supported
MODCP	Requested	Supported
RECLEN	Requested	Supported

Table 13. File attribute list (continued)

Parameter name	Source	Target
RECLENCL	Requested	Supported
RTNCLS[2]	Not requested	PRMFIL
SHDEXS	Not requested	Supported
STRSIZ	Not requested	Supported
TITLE[3]	Requested	Supported

[1] Name formats are server defined. Qualified name if FILCLS is ALTINDF.

[2] Unless the library is QTEMP.

[3] Maximum length of text is 50 characters for data file, 44 for document or folder.

FRCBFF (Force Buffer) Level 2.0

This command forces the data of the referred object to nonvolatile storage.

Parameter name	Source	Target
DCLNAM[1]	Requested	Program defined

[1]Names are implementation defined.

GETDRCEN (Get Directory Entries) Level 2.0

This command gets a list of folders, documents or both. This command is not sent by a source iSeries server.

Parameter name	Source	Target
BGNNAM[1]	N/A	iSeries name
DCLNAM[2]	N/A	Program defined
FILATTRL	N/A	Supported
FILCLS	N/A	DIRECTORY or STRFIL only
FILHDD	N/A	Supported
FILSYS	N/A	Supported
MAXGETCN	N/A	Supported
NAME[1]	N/A	iSeries name

[1] Name formats are server defined.

[2] Names are implementation defined.

The following reply object is possible:

FILAL File attribute list

GETREC (Get Record at Cursor) Level 1.0

This command gets and returns the record indicated by the current cursor position.

Parameter name	Source	Target
DCLNAM[1]	Program defined	Program defined
KEYVALFB	Requested	Supported
RECNBRFB	Requested	Supported
RTNINA[2]	As required	Supported
UPDINT	Not sent	Supported

[1] Names are implementation defined.

[2] Application dependent.

The following reply objects are possible:

RECAL
Record attribute list

RECINA
Inactive record (-1 not supported, maximum =2**15-2) (2 to the power of 15 minus 2.)

RECORD
Fixed length record (maximum length 2**15-2)

GETSTR (Get Substream) Level 2.0 and Level 3.0

This command gets stream data from a document. This command is not sent by a source iSeries server.

Parameter name	Source	Target
DCLNAM[1]	N/A	Program defined
STRLEN	N/A	Supported
STROFF	N/A	Supported
STRPOS	N/A	Supported

[1]Names are implementation defined.

INSRECEF (Insert at EOF) Level 1.0

This command inserts a record at the end of the file.

Parameter name	Source	Target
DCLNAM[1]	Program defined	Program defined
KEYVALFB	Requested	Supported
RECCNT[2]	As required	Supported
RECNBRFB	Requested	Supported
RLSUPD	Always FALSE	Supported
UPDCSR	Not sent	Supported

[1] Names are implementation defined.

[2] Application dependent.

The following command objects are possible:

RECINA
Inactive record (-1 not supported, maximum = 2**15-2) (2 to the power of 15 minus 2)

RECORD
Fixed length record (maximum length 2**15-2)

The following reply objects are possible:

KEYVAL
Key value

RECAL
Record attribute list

RECNBR
Record number

INSRECKY (Insert Record by Key Value) Level 1.0

This command inserts one or more records according to their key values wherever there is space available in the file.

Parameter name	Source	Target
DCLNAM[1]	Program defined	Program defined
RECCNT	As required	Supported
RECNBRFB	Requested	Supported
RLSUPD	Always FALSE	Supported
UPDCSR	Not sent	Supported

[1]Names are implementation defined.

The following command object is possible:

RECORD
Fixed length record (maximum length 2**15-2) (2 to the power of 15 minus 2)

Because the iSeries server does not support variable length records, only the following reply object is possible:

RECNBR
Record number

INSRECNB (Insert Record at Number) Level 1.0

This command inserts one or more records at the position specified by the record number parameter.

Parameter name	Source	Target
DCLNAM[1]	Program defined	Program defined
KEYVALFB	Requested	Supported
RECCNT	As required	Supported
RECNBR	Sent	Supported
UPDCSR	Not sent	Supported

[1]Names are implementation defined.

The following command objects are possible:

RECINA
Inactive record (-1 not supported, maximum = 2**15-2) (2 to the power of 15 minus 2)

RECORD
Fixed length record (maximum length 2**15-2)

The following reply object is possible:

KEYVAL
Key value

LCKFIL (Lock File) Level 1.0 and Level 2.0

This command locks the file for subsequent use by the requester.

Parameter name	Source	Target
FILNAM[1]	Target name	iSeries name
LCKMGRNM	Not used	Ignored
RQSFILLK	Sent	Supported
WAIT	Sent	Supported

[1]Name formats are server defined.

LCKSTR (Lock Substream) Level 2.0 and Level 3.0

This command locks a stream file substream. This command is not sent by a source iSeries server.

Parameter name	Source	Target
DCLNAM[1]	N/A	Program defined
RQSSTRLK	N/A	EXCSTRLK and SHRSTRLK only
STRLOC	N/A	Supported
STROFF	N/A	Supported
WAIT[2]	N/A	Supported

[1] Names are implementation defined.

[2] The WAIT parameter is neither rejected nor performed.

LODRECF (Load Record File) Level 1.0 and Level 2.0

This command puts a whole record file on the target server.

Parameter name	Source	Target
FILNAM[1]	Sent	iSeries name

[1]Name formats are server defined.

The following command objects are possible:

RECAL
Record attribute list

RECCNT
Record count

RECINA
Inactive record (-1 not supported, maximum = 2**15-2) (2 to the power of 15 minus 2)

RECORD
Fixed length record (maximum length 2**15-2)

LODSTRF (Load Stream File) Level 2.0

This command sends a whole stream file from the source server to the target server. This command is sent by a source iSeries server when using the copy stream file HPS API.

Parameter name	Source	Target
FILNAM[1]	Sent	iSeries name

[1]Name formats are server defined.

The following command objects are possible:

STREAM
Stream

STRSIZ
Stream size

Related reference

"Hierarchical file system API support for DDM" on page 40
The hierarchical file system (HFS) APIs and the functions that they support are part of the i5/OS operating system.

LSTFAT (List File Attributes) Level 1.0, Level 2.0, and Level 3.0

This command retrieves selected attributes of a file, document, or folder.

Parameter name	Source	Target
FILATTRL	Sent	Supported
FILNAM[1]	Target name	iSeries name
DCLNAM[2]	Not sent	Supported

[1] Name formats are server defined.

[2] Names are implementation defined.

The following reply object is possible:

FILAL List file attributes reply data

MODREC (Modify Record with Update Intent) Level 1.0

This command changes the record that currently has update intent placed on it without affecting the current cursor position.

Parameter name	Source	Target
ALWMODKY	Sent	Supported
DCLNAM[1]	Sent	Program defined

[1]Names are implementation defined.

The following command object is possible:

RECORD
Fixed length record (maximum length 2**15-2) (2 to the power of 15 minus 2)

OPEN (Open File) Level 1.0 and Level 2.0

This command establishes a logical connection between the using program on the source server and the object on the target server.

Parameter name	Source	Target
ACCINTLS	Sent	Supported
ACCMTHCL	Sent	Supported
DCLNAM[1]	Program defined	Program defined
FILATTRL	Not sent	Supported
FILSHR	Sent	Supported
PRPSHD	Not sent	Supported for stream files only

[1]Names are implementation defined.

OPNDRC (Open Directory) Level 2.0

This command opens a folder on the target server. This command is not sent by a source iSeries server.

Parameter name	Source	Target
ACCMTHCL	N/A	DRCAM only
DCLNAM[1]	N/A	Program defined

[1]Names are implementation defined.

PUTSTR (Put Substream) Level 2.0 and Level 3.0

This command puts stream data into a document. This command is not sent by a source iSeries server.

Parameter name	Source	Target
DCLNAM[1]	N/A	Program defined

Parameter name	Source	Target
STROFF	N/A	Supported
STRPOS	N/A	Supported
[1]Names are implementation defined.		

The following command object is possible:

STREAM
Stream

QRYCD (Query Current Directory) Level 2.0

This command returns the current directory. This command is not sent by a source iSeries server.

Parameter name	Source	Target
AGNNAM	N/A	Ignored

The following reply object is possible:

DRCNAM
Directory name

Note: A directory name length of zero indicates that the root directory is the *current* directory.

QRYSPC (Query Space) Level 2.0

This command returns the amount of space available to a user. This command is not sent by a source iSeries server.

Parameter name	Source	Target
AGNNAM	N/A	Ignored

The following reply object is possible:

QRYSPCRD
Query space reply data

RNMDRC (Rename Directory) Level 2.0

This command renames a folder or database library. The command does not support moving folders, and is not sent by a source iSeries server.

Parameter name	Source	Target
DRCNAM	N/A	iSeries name
NEWDRCNM	N/A	iSeries name
Note: Name formats are server defined. Generic names are not allowed.		

RNMFIL (Rename File) Level 1.0 and Level 2.0

This command changes the name of an existing database file or document and can also be used for moving documents.

Parameter name	Source	Target
FILNAM[1]	Sent	iSeries name
NEWFILNM[2]	Sent	iSeries name

Parameter name	Source	Target

[1] Name formats are server defined. Generic names are allowed for documents only.

[2] Name formats are server defined.

SBMSYSCMD (Submit server Command) Level 4.0

This command submits a server command, in the target control language syntax, to the target server.

Parameter name	Source	Target
SYSCMD[1]	Sent	Supported

[1]Command string to be run.

SETBOF (Set Cursor to Beginning of File) Level 1.0

This command sets the cursor to the beginning-of-file position of the file.

Parameter name	Source	Target
DCLNAM[1]	Program defined	Program defined

[1]Names are implementation defined.

SETEOF (Set Cursor to End of File) Level 1.0

This command sets the cursor to the end-of-file position of the file.

Parameter name	Source	Target
DCLNAM[1]	Program defined	Program defined

[1]Names are implementation defined.

SETFRS (Set Cursor to First Record) Level 1.0

This command sets the cursor to the first record of the file.

Parameter name	Source	Target
BYPINA[1]	As required	Supported
DCLNAM[2]	Program defined	Program defined
HLDCSR	Requested	Supported
KEYVALFB	Requested	Supported
RECNBRFB	Requested	Supported
RTNREC[3]	Sent	Supported
UPDINT[3]	Sent	Supported

[1] Application dependent.

[2] Names are implementation defined.

[3] iSeries server preferred implementation.

The following reply objects are possible:

KEYVAL
Key value

RECAL
Record attribute list

RECINA
Inactive record (-1 not supported, maximum = 2**15-2) (2 to the power of 15 minus 2)

RECNBR
Record number

RECORD
Record

SETKEY (Set Cursor by Key) Level 1.0

This command positions the cursor based on the key value supplied and the relational operator specified for RELOPR.

Parameter name	Source	Target
DCLNAM[1]	Program defined	Program defined
HLDCSR	Requested	Supported
KEYVAL[2]	Max = 2000	Max = 2000
KEYVALFB	Requested	Supported
RECNBRFB	Requested	Supported
RELOPR	Sent	Supported
RTNREC[3]	Sent	Supported
UPDINT[3]	Sent	Supported

[1] Names are implementation defined.

[2] Maximum key size allowed by an iSeries server.

[3] iSeries server preferred implementation.

The following reply objects are possible:

KEYVAL
Key value

RECAL
Record attribute list

RECNBR
Record number

RECORD
Record

SETKEYFR (Set Cursor to First Record in Key Sequence) Level 1.0

This command sets the cursor to the first record in the key sequence.

Parameter name	Source	Target
DCLNAM[1]	Program defined	Program defined
HLDCSR	Requested	Supported
KEYVALFB	Requested	Supported
RECNBRFB	Requested	Supported
RTNREC[2]	Sent	Supported
UPDINT[2]	Sent	Supported

[1] Names are implementation defined.

[2] iSeries server preferred implementation.

The following reply objects are possible:

KEYVAL
Key value

RECAL
Record attribute list

RECNBR
Record number

RECORD
Record

SETKEYLM (Set Key Limits) Level 1.0

This command sets the limits of the key values for subsequent SETKEYNX and SETNXTKE commands. This command is not sent by a source iSeries server.

Parameter name	Source	Target
DCLNAM[1]	N/A	Program defined
KEYHLM[2]	N/A	Supported
KEYLLM[2]	N/A	Supported

[1] Names are implementation defined.

[2] Application dependent.

SETKEYLS (Set Cursor to Last Record in Key Sequence) Level 1.0

This command sets the cursor to the last record of the file in key sequence order.

Parameter name	Source	Target
DCLNAM[1]	Program defined	Program defined
HLDCSR	Requested	Supported
KEYVALFB	Requested	Supported
RECNBRFB	Requested	Supported
RTNREC[2]	Sent	Supported
UPDINT[2]	Sent	Supported

[1] Names are implementation defined.

[2] iSeries server preferred implementation.

The following reply objects are possible:

KEYVAL
Key value

RECAL
Record attribute list

RECNBR
Record number

RECORD
Record

SETKEYNX (Set Cursor to Next Record in Key Sequence) Level 1.0

This command sets the cursor to the next record of the file in the key sequence order following the record currently indicated by the cursor.

Parameter name	Source	Target
BYPDMG[1]	Not sent	Supported
DCLNAM[2]	Program defined	Program defined
HLDCSR	Requested	Supported

Parameter name	Source	Target
KEYVALFB	Requested	Supported
RECCNT[1]	As required	Supported
RECNBRFB	Requested	Supported
RTNREC[3]	Sent	Supported
UPDINT[3]	Sent	Supported

[1] Application dependent.

[2] Names are implementation defined.

[3] iSeries server preferred implementation.

The following reply objects are possible:

KEYVAL
Key value

RECAL
Record attribute list

RECNBR
Record number

RECORD
Record

SETKEYPR (Set Cursor to Previous Record in Key Sequence) Level 1.0

This command sets the cursor to the previous record of the file in the key sequence order preceding the record currently indicated by the cursor.

Parameter name	Source	Target
DCLNAM[1]	Program defined	Program defined
HLDCSR	Requested	Supported
KEYVALFB	Requested	Supported
RECCNT[2]	As required	Supported
RECNBRFB	Requested	Supported
RTNREC[3]	Sent	Supported
UPDINT[3]	Sent	Supported

[1] Names are implementation defined.

[2] Application dependent.

[3] iSeries server preferred implementation.

The following reply objects are possible:

KEYVAL
Key value

RECAL
Record attribute list

RECNBR
Record number

RECORD
Record

SETLST (Set Cursor to Last Record) Level 1.0

This command sets the cursor to the last record of the file.

Parameter name	Source	Target
BYPINA[1]	As required	Supported
DCLNAM[2]	Program defined	Program defined
HLDCSR	Requested	Supported
KEYVALFB	Requested	Supported
RECNBRFB	Requested	Supported
RTNREC[3]	Sent	Supported
UPDINT[3]	Sent	Supported

[1] Application dependent.

[2] Names are implementation defined.

[3] iSeries server preferred implementation.

The following reply objects are possible:

KEYVAL
Key value

RECAL
Record attribute list

RECINA
Inactive record (-1 not supported, maximum = 2**15-2)

RECNBR
Record number

RECORD
Record

SETMNS (Set Cursor Minus) Level 1.0

This command sets the cursor to the record number of the file indicated by the cursor minus the number of record positions specified by CSRDSP.

Parameter name	Source	Target
ALWINA[1]	As required	Supported
CSRDSP[1]	Sent	Supported
DCLNAM[2]	Program defined	Program defined
HLDCSR	Requested	Supported
KEYVALFB	Requested	Supported
RECNBRFB	Requested	Supported
RTNINA[1]	As required	Supported
RTNREC[3]	Sent	Supported
UPDINT[3]	Sent	Supported

[1] Application dependent.

[2] Names are implementation defined.

[3] iSeries server preferred implementation.

The following reply objects are possible:

KEYVAL
Key value

RECAL
Record attribute list

RECINA
Inactive record (-1 not supported, maximum = 2**15-2) (2 to the power of 15 minus 2)

RECNBR
Record number

RECORD
Record

SETNBR (Set Cursor to Record Number) Level 1.0

This command sets the cursor to the record of the file indicated by the record number specified by RECNBR.

Parameter name	Source	Target
ALWINA[1]	As required	Supported
DCLNAM[2]	Program defined	Program defined
HLDCSR	Requested	Supported
KEYVALFB	Requested	Supported
RECNBR	Sent	Supported
RTNINA[1]	As required	Supported
RTNREC[3]	Sent	Supported
UPDINT[3]	Sent	Supported

[1] Application dependent.

[2] Names are implementation defined.

[3] iSeries server preferred implementation.

The following reply objects are possible:

KEYVAL
Key value

RECAL
Record attribute list

RECINA
Inactive record (-1 not supported, maximum = 2**15-2) (2 to the power of 15 minus 2)

RECORD
Record

SETNXT (Set Cursor to Next Number) Level 1.0

This command sets the cursor to the next record of the file with a record number one greater than the current cursor position.

Parameter name	Source	Target
BYPDMG[1]	Not sent	Supported
BYPINA[1]	As required	Supported
DCLNAM[2]	Program defined	Program defined
HLDCSR	Requested	Supported
KEYVALFB	Requested	Supported
RECCNT[1]	As required	Supported

Parameter name	Source	Target
RECNBRFB[1]	As required	Supported
RTNREC[3]	Sent	Supported
UPDINT[3]	Sent	Supported

[1] Application dependent.

[2] Names are implementation defined.

[3] iSeries server preferred implementation.

The following reply objects are possible:

KEYVAL
Key value

RECAL
Record attribute list

RECINA
Inactive record (-1 not supported, maximum = 2**15-2) (2 to the power of 15 minus 2)

RECNBR
Record number

RECORD
Record

SETNXTKE (Set Cursor to Next Record in Key Sequence with a Key Equal to Value Specified) Level 1.0

This command positions the cursor to the next record in the key sequence if the key field of that record has a value equal to the value specified in the KEYVAL parameter. This command is not sent by a source iSeries server.

Parameter name	Source	Target
DCLNAM[1]	N/A	Program defined
HLDCSR	N/A	Supported
KEYVAL[2]	N/A	Max = 2000
KEYVALFB	N/A	Supported
RECNBRFB	N/A	Supported
RTNREC[3]	N/A	Supported
UPDINT[3]	N/A	Supported

[1] Names are implementation defined.

[2] Maximum key size allowed by an iSeries server.

[3] iSeries server preferred implementation.

The following reply objects are possible:

KEYVAL
Key value

RECAL
Record attribute list

RECNBR
Record number

RECORD
Record

SETPLS (Set Cursor Plus) Level 1.0

This command sets the cursor to the record number of the file indicated by the cursor plus the integer number of records specified by CSRDSP.

Parameter name	Source	Target
ALWINA[1]	As required	Supported
CSRDSP[1]	Sent	Supported
DCLNAM[2]	Program defined	Program defined
HLDCSR	Requested	Supported
KEYVALFB	Requested	Supported
RECNBRFB	Requested	Supported
RTNINA[1]	As required	Supported
RTNREC[3]	Sent	Supported
UPDINT[3]	Sent	Supported

[1] Application dependent.

[2] Names are implementation defined.

[3] iSeries server preferred implementation.

The following reply objects are possible:

KEYVAL
Key value

RECAL
Record attribute list

RECINA
Inactive record (-1 not supported, maximum = 2**15-2) (2 to the power of 15 minus 2)

RECNBR
Record number

RECORD
Record

SETPRV (Set Cursor to Previous Record) Level 1.0

This command sets the cursor to the record of the file with a record number one less than the current cursor position.

Parameter name	Source	Target
BYPINA[1]	As required	Supported
DCLNAM[2]	Program defined	Program defined
HLDCSR	Requested	Supported
KEYVALFB	Requested	Supported
RECCNT[1]	As required	Supported
RECNBRFB	Requested	Supported
RTNREC[3]	Sent	Supported
UPDINT[3]	Sent	Supported

[1] Application dependent.

[2] Names are implementation defined.

[3] iSeries server preferred implementation.

The following reply objects are possible:

RECAL
Record attribute list

RECINA
Inactive record (-1 not supported, maximum = 2 **15-2)

RECNBR
Record number

RECORD
Record

SETUPDKY (Set Update Intent by Key Value) Level 1.0

This command places an update intent on the record that has a key value equal to the key value specified by KEYVAL.

Parameter name	Source	Target
DCLNAM[1]	Program defined	Program defined
KEYVAL[2]	Max = 2000	Max = 2000
KEYVALFB	Requested	Supported
RECNBRFB	Requested	Supported
RTNREC[3]	Sent	Supported

[1] Names are implementation defined.

[2] Maximum key size allowed by an iSeries server.

[3] Only RTNREC(FALSE) is supported.

The following reply objects are possible:

KEYVAL
Key value

RECAL
Record attribute list

RECNBR
Record number

RECORD
Record

SETUPDNB (Set Update Intent by Record Number) Level 1.0

This command places an update intent on the record of the file indicated by the record number specified by RECNBR.

Parameter name	Source	Target
ALWINA[1]	As required	Supported
DCLNAM[2]	Program defined	Program defined
KEYVALFB	Requested	Supported
RECNBR	Sent	Supported
RTNINA[1]	As required	Supported
RTNREC[3]	Sent	Supported

[1] Application dependent.

[2] Names are implementation defined.

[3] Only RTNREC(FALSE) is supported.

The following reply objects are possible:

KEYVAL
Key value

RECAL
Record attribute list

RECINA
Inactive record (-1 not supported, maximum = 2**15-2) (2 to the power of 15 minus 2)

RECORD
Record

ULDRECF (Unload Record File) Level 1.0

This command sends records from a target record file to the source.

Parameter name	Source	Target
ACCDORD[1]	Sent	Supported
BYPDMG[1]	Sent	Supported
FILNAM[2]	Sent	iSeries name
RTNINA[1]	Sent	Supported

[1] Application dependent.

[2] Name formats are server defined.

The following reply objects are possible:

RECAL
Record attribute list

RECCNT
Record count

RECINA
Inactive record (-1 not supported, maximum = 2 **15-2) (2 to the power of 15 minus 2)

RECORD
Record

ULDSTRF (Unload Stream File) Level 2.0

This command sends a document from the target to the source. This command is sent by a source iSeries server when using the Copy Stream File (QHFCPYSF) HFS API.

Parameter name	Source	Target
BYPDMG	Not sent	FALSE only
FILNAM[1]	Sent	iSeries name
STRORD	Not sent	Supported

[1]Name formats are server defined.

The following reply objects are possible:

STREAM
Stream

STRPOS
Stream position

STRSIZ
Stream size

Related reference

"Hierarchical file system API support for DDM" on page 40
The hierarchical file system (HFS) APIs and the functions that they support are part of the i5/OS operating system.

UNLFIL (Unlock File) Level 1.0 and Level 2.0

This command releases explicit file locks held by the requester on the file.

Parameter name	Source	Target
FILNAM[1]	Target name	iSeries name
LCKMGRNM	Not sent	Ignored
RLSFILLK	Sent	Supported

[1]Name formats are server defined.

UNLIMPLK (Unlock Implicit Record Lock) Level 1.0

This command releases all implicit record locks currently held by the cursor.

Parameter name	Source	Target
DCLNAM[1]	Program defined	Program defined

[1]Names are implementation defined.

UNLSTR (Unlock Substreams) Level 2.0 and Level 3.0

This command unlocks a stream file substream. This command is not sent by a source iSeries server.

Parameter name	Source	Target
DCLNAM[1]	N/A	Program defined
STRLOC	N/A	Supported

[1]Names are implementation defined.

User profile authority

The user profile associated with target iSeries server jobs must be authorized to the equivalent CL commands before the DDM command can be processed. The target job's user profile must be authorized to use the CL commands listed here before DDM requests can be processed.

Table 14. User profile authority CL commands

DDM command received	DDM command description	Object type	Authorized CL command
CHGDRC	Change Current Directory	FLR	NONE
CHGFAT	Change File Attributes	PFILE LF DOC/FLR	CHGPF CHGLF NONE
CLOSE	Close File	FILE DOC	NONE [1] NONE
CLRFIL	Clear File	FILE DOC	NONE NONE
CLSDRC	Close Directory	FLR	NONE
CPYFIL	Copy File	DOC	NONE
CRTAIF	Create Alternate Index File	LF	CRTLF
CRTDIRF	Create Direct File	PF	CRTPF
CRTKEYF	Create Key File	PF	CRTPF
CRTSEQF	Create Sequential File	PF	CRTPF
CRTSTRF	Create Stream File	DOC	NONE
CRTDRC	Create Directory	LIB FLR	CRTLIB CRTFLR

Table 14. User profile authority CL commands (continued)

DDM command received	DDM command description	Object type	Authorized CL command
DELFIL	Delete File	FILE DOC	DLTF NONE
DELDRC	Delete Directory	LIB FLR	DLTLIB NONE
GETDRCEN	Get Directory Entry	DOC/FLR	NONE
LCKFIL	Lock File	FILE	ALCOBJ
LODRECF	Load (Put) Records to File	FILE	NONE [2]
LSTFAT	List File Attributes	FILE DOC/FLR	NONE [3] NONE
OPEN	Open File	FILE DOC	NONE [1] NONE
OPENDRC	Open Directory	FLR	NONE
QRYSPC	Query Space Available to User	USRPRF	NONE [4]
RNMDRC	Rename Directory	FLR LIB	NONE RNMOBJ
RNMFIL	Rename File	FILE DOC MBR	RNMOBJ NONE RNMM
UNLFIL	Unlock File	FILE	NONE [5]
ULDRECF	Unload Records From File	FILE	NONE [2]

[1] Authorization to a command is not verified because there are means other than using a command interface by which iSeries users can open and close files.

[2] Command authorization is not verified because there is not a direct, one-to-one mapping between a CL command and the DDM LODRECF/ULDRECF command.

[3] Authorization to the DSPFD and DSPFFD commands is not verified because it cannot be determined which command should be verified. In addition, the conditions under which the DDM command was issued by the source server are not known.

[4] The space available to a user can be obtained by issuing the DSPUSRPRF command, but this is only a small piece of the data available through the use of this command.

[5] Authorization to the CL DLCOBJ command is not checked because if the remote user was able to allocate files, DDM must be able to deallocate them.

The following table is an explanation of the object type codes used in the preceding table.

Table 15. Object type codes definition

Object type	Object type definition
DOC	Document
FLR	Folder
PF	Physical file
LF	Logical file
LIB	Library
MBR	Member
SRCF	Source physical file
USRPRF	User profile

iSeries server-to-CICS considerations with DDM

This topic describes programming considerations for accessing CICS remote files with i5/OS DDM.

Note: A System/370™ host must have installed CICS/OS/VS Version 1.7 or later and CICS/DDM Version 1.1.

iSeries languages, utilities, and licensed programs

The iSeries languages, utilities, and licensed programs in this topic can access remote CICS files.

- Programs written in the following languages on an iSeries server can access remote CICS files:

  ILE C programming language
  See "ILE C considerations" on page 202.

  CL See "iSeries CL considerations" on page 196.

  ILE COBOL programming language
  See "ILE COBOL considerations" on page 200.

  See "PL/I considerations" on page 198.

  ILE RPG programming language
  See "ILE RPG considerations" on page 203.
- Programs written in BASIC might cause results that cannot be predicted when accessing remote CICS files.
- iSeries Query can access the remote entry sequence data set (ESDS), relative record data set (RRDS), and key sequence data set (KSDS). However, iSeries Query cannot access virtual storage access method (VSAM) files through DDM.
- The licensed program iSeries Access Family might cause results that cannot be predicted when accessing remote CICS files.

Note: Some of the high-level languages provide access to the server database I/O feedback area. When accessing a remote VSAM RRDS, this area will contain the relative record number. However, when accessing other types of VSAM data sets, the relative record number is not known and a value of -1 is returned as the relative record number.

Additional considerations might apply when accessing CICS files which are to be read or written by a System/370 host due to the way a System/370 host stores data. For example, the System/370 host representation of a floating point number is different from the iSeries server representation of a floating point number.

CRTDDMF (Create DDM File) considerations

For applications running on an iSeries server to access remote files, the programmer must use the CRTDDMF command to create an object called a DDM file.

The ACCMTH parameter of this command shows which DDM access method should be used when opening the remote file. If *RMTFILE is used, i5/OS DDM selects an access method that is compatible with:

- The type of VSAM data set being accessed
- The access methods supported by CICS/DDM for the VSAM data set

The table below shows how the possible values for the ACCMTH parameter correspond to VSAM data sets.

Table 16. ACCMTH parameter of iSeries CRTDDMF command

ACCMTH parameter values	VSAM data set organization			
	ESDS	**RRDS**	**KSDS**	**VSAM Path**
*ARRIVAL	R	R	E	E
*KEYED	E	E	R	R
*BOTH	E	O	O	O
*RANDOM	E	O	O	O

Table 16. ACCMTH parameter of iSeries CRTDDMF command (continued)

ACCMTH parameter values	VSAM data set organization			
	ESDS	**RRDS**	**KSDS**	**VSAM Path**
*SEQUENTIAL	R	O	O	O
*COMBINED	E	O	E	E

Where:

R Shows the parameter is required for accessing the VSAM data set.

O Shows the parameter is optional for accessing the VSAM data set.

E Shows the parameter causes an i5/OS message.

To improve performance, the iSeries user might want to supply values other than *RMTFILE for the ACCMTH parameter. To avoid server messages, use the values specified in Table 16 on page 195 when accessing remote VSAM data sets.

The value specified for the RMTFILE file parameter must be the same as the value specified for the DATASET parameter of the CICS DFHFCT macro when the VSAM data set is defined to the CICS system.

iSeries CL considerations

Besides the information in this topic collection, consider these topics when using iSeries CL commands to access VSAM data sets on a remote CICS system.

Note: Commands that do not appear in the following topics have no considerations besides those stated in this topic collection.

ALCOBJ (Allocate Object):

Unless the CICS system programmer replaces the CICS/DDM-supplied exclusive file lock program with a special version of the program, a lock condition value of *SHRRD or *SHRUPD must be used when using the Allocate Object (ALCOBJ) command to allocate a remote VSAM data set. All other lock condition values result in a server message on the iSeries server.

CLRPFM (Clear Physical File Member):

The Clear Physical File Member (CLRPFM) command cannot clear a VSAM data set on a remote CICS system.

CPYF (Copy File):

The Copy File (CPYF) command can access remote VSAM data sets defined to a CICS system.

However, listed here is what you need to pay attention to:

- When the TOFILE parameter is a remote VSAM data set:
  - The CRTFILE parameter must have a value of *NO.
  - The MBROPT parameter must have a value of *ADD.
  - The FMTOPT parameter must have a value of *NOCHK.
- When the TOFILE parameter is a remote VSAM ESDS or KSDS, the COMPRESS parameter must have a value of *YES.

CPYTOTAP, CPYFRMTAP and CPYSPLF commands:

The Copy to Tape (CPYTOTAP), and the Copy from Tape (CPYFRMTAP) commands access remote VSAM data sets defined to a CICS system.

However, *ADD must be used for the MBROPT parameter. The Copy Spool File (CPYSPLF) command cannot access remote VSAM data sets.

DLCOBJ (Deallocate Object):

The Deallocate Object (DLCOBJ) command can release any lock successfully acquired for a remote VSAM data set.

DSPFD and DSPFFD commands:

The Display File Description (DSPFD) and Display File Field Description (DSPFFD) commands can display information about a remote VSAM data set.

However, the information for many of the fields is not available to CICS/DDM and is not returned to i5/OS DDM. Refer to the following table for the fields that CICS/DDM does not return:

Table 17. CICS/DDM file and file field descriptions

Field	Value
Creation date	Zeros
Current number of records	Zeros
Data space in bytes	Zeros
File level identifier	Zeros
File text description	Zeros
Format level identifier	Blanks
Last change date and time	Zeros
Member creation date	Zeros
Member expiration date	*NONE
Member level identifier	Zeros
Member size	*NOMAX
Number of deleted records	Zeros
Text description	Blanks
Total deleted records	Zeros
Total member size	Zeros
Total records	Zeros

Note: The values displayed do not represent actual data about the file. They act as default values for the information that CICS/DDM does not return.

When the type of file is logical, the information displayed shows that unique keys are not required. In fact, CICS/DDM does not know whether unique keys are required or not.

Sometimes, the iSeries user needs to know the type of VSAM data set being accessed. By using the following information, which is displayed by the DSPFD command, it is possible for the iSeries user to make this decision:

- If the type of file is logical, the VSAM data set is a VSAM path.
- If the type of file is physical and the access path is keyed, the VSAM data set is KSDS.
- In all other cases, the VSAM data set is either an RRDS or an ESDS. If the iSeries user must know whether it is an RRDS or an ESDS, the CICS system programmer should be contacted.

DSPPFM (Display Physical File Member):

The Display Physical File Member (DSPPFM) command can be used to access remote relative record data set (RRDS). It does not work for other types of VSAM data sets.

OPNDBF (Open Database File):

The Open Database File (OPNDBF) command can open a remote VSAM data set.

However, a server message is created on the iSeries server if *ARRIVAL is used for the ACCPTH parameter and the remote data set is a VSAM KSDS or a VSAM path.

OVRDBF (Override with Database File):

The Override with Database File (OVRDBF) command can override a local database file with a remote VSAM data set.

However, the following items must be considered:

- A POSITION value of *RRN works when the remote VSAM data set is an RRDS. For all other types of VSAM data sets, *RRN causes a server message on the iSeries server.
- A POSITION value of *KEYB or *KEYBE causes a server message on the iSeries server when the remote file is a VSAM path.
- Unless the CICS system programmer replaces the CICS/DDM-supplied exclusive file lock program, the RCDFMTLCK parameter must have a lock condition value of *SHRRD or *SHRUPD. All other lock condition values result in a server message on the iSeries server.
- CICS/DDM does not return the actual expiration date of the file to i5/OS DDM. Instead, it returns a special value that indicates the expiration date is not known. This is true even if the EXPCHK parameter has a value of *YES.

RCVNETF (Receive Network File):

The Receive Network File (RCVNETF) command can access a VSAM data set defined to a remote CICS system. However, the MBROPT parameter must have a value of *ADD.

Language considerations for iSeries server and CICS

iSeries application programmers using ILE COBOL, ILE C, iSeries System/36-Compatible RPG II, or ILE RPG languages should be aware of the information in these topics.

PL/I considerations

These topics summarize the limitations that exist when using PL/I to access remote VSAM data sets from an iSeries server. These limitations should be considered in addition to those already stated in this topic collection.

PL/I open file requests:

The i5/OS DDM user can access remote CICS files with PL/I programs.

When opening the file with the RECORD file attribute, the program must use the file attributes specified in the table below. By noting the values that appear in this table, you can determine the difference between accessing an iSeries database file and a remote VSAM data set.

Note: Remote files can also be opened with the PL/I STREAM file attribute. However, if the STREAM file attribute is used to open a VSAM KSDS, a server message occurs. This happens because records in a VSAM KSDS cannot be processed in arrival sequence.

Unless the CICS system has replaced the CICS/DDM exclusive file locking program, you cannot use the EXCL and EXCLRD file locking options for the ENVIRONMENT parameter when opening a remote VSAM data set.

Table 18. PL/I file attributes

PL/I file attributes	VSAM data set organization			
	ESDS	**RRDS**	**KSDS**	**VSAM Path**
SEQUENTIAL	R	O	O	O
DIRECT	E	O	O	O
SEQL KEYED	E	O	O	O
INPUT	O	O	O	O
OUTPUT	O	O	O	E
UPDATE	O	O	E	E
CONSECUTIVE	R	R	E	E
INDEXED	–	–	R	R

Where:

R Shows the attribute is required for accessing the VSAM data set.

O Shows the attribute is optional for accessing the VSAM data set.

E Shows the attribute is allowed by PL/I but the open fails when accessing the VSAM data set.

– Shows the option is valid for keyed files only.

Related reference

"PL/I input/output requests"
This topic discusses the types of PL/I input/output requests and their restrictions.

PL/I input/output requests:

This topic discusses the types of PL/I input/output requests and their restrictions.

Read requests

- A KEYSEARCH parameter value of BEFORE or EQLBFR is not supported by CICS/DDM when accessing a VSAM path. However, these parameter values are supported when accessing a VSAM KSDS.
- A POSITION parameter value of PREVIOUS and LAST is not supported when accessing a VSAM path. However, these parameter values are supported when accessing a VSAM KSDS.
- Because the DIRECT or SEQUENTIAL KEYED attributes cannot be used to open a VSAM ESDS, it is not possible to access records by relative record number or to have the relative record number returned by using the KEY and KEYTO parameters.
- Because VSAM KSDS and VSAM alternate indexes are always defined as a single key field, the NBRKEYFLDS parameter should not be used.

Write requests

- The KEYFROM parameter does not work when writing a record to a VSAM RRDS.
- The WRITE request does not work when writing a record to VSAM KSDS that already contains a record with the same key value.
- Because the OUTPUT or UPDATE file attribute cannot be used to open a VSAM path, it is not possible to write records to a VSAM path. Instead, the application program must write the record by using the base data set of the VSAM path.

Rewrite requests

- The REWRITE does not work if rewriting a record of a VSAM KSDS when the key value of the record is changed.
- Because the UPDATE file attribute cannot be used to open a VSAM path, it is not possible to rewrite records in a VSAM path. Instead, the application program must rewrite the record by using the base data set of the VSAM path.

Delete requests

- The DELETE does not work when deleting the record of a VSAM ESDS.
- Because the UPDATE file attribute cannot be used to open a VSAM path, it is not possible to delete records in a VSAM path. Instead, the application program must delete the records by using the base data set of the VSAM path. However, if the base data set of the VSAM path is a VSAM ESDS, the DELETE does not work.

Related reference

"PL/I open file requests" on page 198
The i5/OS DDM user can access remote CICS files with PL/I programs.

ILE COBOL considerations

These topics summarize the limitations that exist when using the ILE COBOL programming language to access remote VSAM data sets from an iSeries server.

Unless otherwise stated, these limitations apply to both the System/36 and the iSeries server. These limitations are in addition to those already stated in this topic collection.

ILE COBOL SELECT clause:

iSeries users can access remote CICS files with the ILE COBOL programming language.

However, the ILE COBOL SELECT clause must use the file organizations and access methods specified in the following table.

Table 19. ILE COBOL file organizations and access methods

ILE COBOL programming language	VSAM data set organization				
Program-given access methods	**ESDS**	**RRDS**	**KSDS**	**VSAM Path**	**Program-given organization**
Sequential	Sequential	X	X	E	E
Relative					
	Sequential	E	X	E	E
	Random	E	X	E	E
	Dynamic	E	X	E	E
Indexed					
	Sequential	–	–	X	X
	Random	–	–	X	X
	Dynamic	–	–	X	X

Table 19. ILE COBOL file organizations and access methods (continued)

ILE COBOL programming language	VSAM data set organization				
Program-given access methods	**ESDS**	**RRDS**	**KSDS**	**VSAM Path**	**Program-given organization**

Where:

X Shows the access method is allowed.

E Shows that ILE COBOL programming language allows the access method but that the open fails when accessing the VSAM data set. An iSeries message is created.

– Shows the option is never valid for nonkeyed files. An iSeries message occurs whenever indexed file organization is selected for any nonkeyed file. This is true even when the file is a local file.

Notes:

1. When accessing a VSAM path, the WITH DUPLICATE phrase should be used.
2. When accessing a VSAM KSDS, the WITH DUPLICATE phrase should not be used.

ILE COBOL statements:

These topics describe considerations you should be aware of when using ILE COBOL statements to access remote VSAM data sets.

ILE COBOL OPEN statement

When accessing remote CICS files, the ILE COBOL OPEN statement must use the open modes that are specified in the following table.

Table 20. Use ILE COBOL programming language to open a CICS file

ILE COBOL open mode	VSAM data set organization			
	ESDS	**RRDS**	**KSDS**	**VSAM Path**
Input	X	X	X	X
Output	E	E	E	E
Input/Output	X	X	X	E
Extend	X	–	–	–

Where:

X Shows the open mode is allowed.

E Shows the open mode is allowed by ILE COBOL programming language but that the open fails when accessing the VSAM data set. A message occurs on an iSeries server.

– Shows the open mode is not applicable.

ILE COBOL READ statement

- The PRIOR phrase and the LAST phrase are not supported by CICS/DDM when accessing a VSAM path. It is supported when accessing a VSAM KSDS.
- Because the RELATIVE file organization can only be used to open a VSAM RRDS, it is not possible to access records by relative record number or to have the relative record number returned from the remote file unless the remote file is a VSAM RRDS.

ILE COBOL WRITE statement

- The WRITE statement does not work when the ILE COBOL program is running on an iSeries server and the file was opened with a RELATIVE file organization.
- The WRITE statement does not work when writing a record to a VSAM KSDS and the data set already contains a record with the same key value.
- Because the input/output and output open modes cannot be used to open a VSAM path, it is not possible to write records to a VSAM path. Instead, the application program must write the record by using the base data set of the VSAM path.

ILE COBOL REWRITE statement

- The REWRITE statement does not work when rewriting the record of a VSAM KSDS and the key value of the record is changed.
- Because the input/output open mode cannot be used to open a VSAM path, it is not possible to rewrite records in a VSAM path. Instead, the application program must rewrite the record by using the base data set of the VSAM path.

ILE COBOL START statement

- The START statement does work if the open mode is INPUT.

ILE COBOL DELETE statement

- Because the sequential file organization must be used to open a VSAM ESDS, it is not possible to delete records in a VSAM ESDS.
- Because the input/output open mode cannot be used to open a VSAM path, it is not possible to delete records in a VSAM path. Instead, the application program must delete the record by using the base data set of the VSAM path. However, if the base data set of the VSAM path is a VSAM ESDS, the DELETE does not work.

ILE C considerations

These topics summarize considerations for using the ILE C programming language to access remote VSAM data sets from an iSeries server.

ILE C open considerations

Because ILE C programming language supports only sequential I/O, open operations will fail if KSDS or VSAM paths are opened.

Open mode considerations

This topic shows the open mode considerations when using ILE C programming language.

Table 21. Use ILE C programming language to open a CICS file

	VSAM data set organization			
ILE C open mode	**ESDS**	**RRDS**	**KSDS**	**VSAM Path**
r, rb	X	X	X	X
w, wb	E	E	E	E
w+, wb+, w+b, a+, ab+, a+b, r+, rb+, r+b, a, ab	X	X	X	E
a, ab	X	—	—	—

Table 21. Use ILE C programming language to open a CICS file (continued)

ILE C open mode	VSAM data set organization			
	ESDS	**RRDS**	**KSDS**	**VSAM Path**
Where:				
X Open mode is allowed.				
E Open mode is allowed by ILE C programming language, but the open fails when accessing the VSAM data set.				
— Open mode is not applicable.				

ILE RPG considerations

These topics summarize the limitations that exist when using the iSeries System/36-Compatible RPG II or the ILE RPG programming language on an iSeries server to access remote VSAM data sets.

These limitations are in addition to those already stated in this topic collection.

File description specifications:

iSeries users can access remote VSAM data sets with iSeries System/36-Compatible RPG II or ILE RPG programming language.

However, not all ILE RPG processing methods selected by the file description specifications can access remote VSAM data sets. Refer to the following tables to determine which file description specifications to use. If a file description specification other than what appears in these tables is used, CICS/DDM rejects the request for opening the file.

Table 22. ILE RPG processing methods for remote VSAM ESDS

Type of processing	Column number						
	15	**16**	**19**	**28**	**31**	**32**	**66**
Consecutive	I I I I U U U	P S T D P S D	F F F F F F F				
Add Records Only	O						A

Table 23. ILE RPG processing methods for remote VSAM RRDS

Type of processing	Column number						
	15	**16**	**19**	**28**	**31**	**32**	**66**
Consecutive	I I I U U U	P S D P S D	F F F F F F				

Table 23. ILE RPG processing methods for remote VSAM RRDS (continued)

Type of processing	Column number						
	15	**16**	**19**	**28**	**31**	**32**	**66**
Random by Chain		C	F	R			A
See note.	I	C	F	R			A
See note.	I	C	F	R			
	U	C	F	R			
	U						
Random by Addrout							
	I	P	F	R	I		
	I	S	F	R	I		
	I	F	F	R	I		
	U	P	F	R	I		
	U	S	F				
	U	F	F				
Consecutive, Random or both	I	F	F				A
See note.	I	F	F				A
See note.	U	F	F				
	U	F	F				

Note: A **K** must be used in column 53 and **RECNO** must be in columns 54 through 59 to indicate relative record number processing.

Table 24. ILE RPG processing methods for VSAM KSDS

Type of processing	Column number						
	15	**16**	**19**	**28**	**31**	**32**	**66**
Sequential	I	P	F	L		I	A
By key, no add	I	S	F	L		I	A
By key, no add	I	D	F	L		I	A
By key, no add	I	P	F	L		I	A
By key, with add	I	S	F	L		I	A
By key, with add	I	D	F	L		I	A
By key, with add	U	P	F	L		I	A
By key, no add	U	S	F	L		I	A
By key, no add	U	D	F	L		I	
By key, no add	U	P	F			I	
By key, with add	U	S	F			I	
By key, with add	U	D	F			I	
By key, with add	I	P	F			I	
By limits	I	S	F			I	
By limits	I	F	F			I	
By limits	U	D	F			I	
By limits	U	P	F			I	
By limits	U	D	F			I	
By limits	U	F	F			I	
By limits	I	F	F			I	
By limits, adds	I	F	F			I	
By limits, adds							

Table 24. ILE RPG processing methods for VSAM KSDS (continued)

Type of processing	Column number						
	15	**16**	**19**	**28**	**31**	**32**	**66**
Random by Chain No adds With adds No adds With adds	I I U U	C C C C	F F F F	R R R R		I I I I	A A
Random by Addrout	I I U U	P S P S	F F F F	R R R R	I I I I	I I I I	
Sequential, Random or both By key, no add By key, with add By key, no add By key, no add	I I U U	F F F F	F F F F			I I I I	A A
Add Records Only	O					I	A

Table 25. Processing methods for remote VSAM paths

Type of processing	Column number						
	15	**16**	**19**	**28**	**31**	**32**	**66**
Sequential By key, no add By key, no add By key, no add By limits By limits By limits By limits	I I I I I I I	P S D P S F D	F F F F F F F	L L L L		I I I I I I I	
Random by Chain No adds	I	C	F	R		I	
Random by Addrout	I I	P S	F F	R R	I I	I I	
Sequential, Random or both By key, no add	I	F	F			I	

ILE RPG input/output operations:

Be aware of these restrictions when accessing remote VSAM data sets.

- Records can be read or added by relative record number only when the remote VSAM data set is an RRDS. Random processing by relative record number can be used only when processing a VSAM RRDS.
- A request to delete a record in an ESDS does not work because ESDS is never delete-capable.

- The processing method cannot use the update or output specification in column 15 when the remote VSAM data set is a VSAM path. Instead, all add, update, or delete record requests must be made by using the base data set of the VSAM path. However, if the base data set of the VSAM path is a VSAM ESDS, the DELETE does not work.
- The READP operation code cannot be used to read the previous record in a VSAM path.
- Because VSAM KSDS does not allow duplicate keys, a request to add a record that duplicates the key of an existing record in a KSDS does not work.
- When accessing a KSDS, an update request that changes the key value of the record does not work.
- For ILE RPG programming language, *HIVAL can be used to obtain the last record of a remote KSDS. However, *HIVAL does not work when accessing a VSAM path.

Use DDM on the iSeries server versus other IBM systems

This topic describes the DDM differences between iSeries server and System/36, and iSeries server and System/38.

Related concepts


System/38 Environment Programming PDF

iSeries server and System/36 DDM differences

This topic consists of a list of differences between the iSeries server and System/36.

- The network resource directory (NRD) procedures are not supported on the iSeries server.
  - The System/36 NRD used one or more libraries containing DDM files on the iSeries server. One System/36 NRD entry is equivalent to one DDM file on the iSeries server.
    - Use the Work with DDM Files (WRKDDMF) command in place of the EDITNRD and DELNRD procedures.
    - Use the Display DDM Files (DSPDDMF) command in place of the LISTNRD procedure.
    - Use the Restore Object (RSTOBJ) command in place of the RESTNRD procedure.
    - Use the Save Object (SAVOBJ) command in place of the SAVNRD procedure.
  - If an NRD entry on the System/36 refers to a remote file, and a SAVE or RESTORE procedure is requested, the System/36 saves or restores the data of the remote file. The iSeries Save Object (SAVOBJ) or Restore Object (RSTOBJ) command saves or restores only the definition of the DDM file, not the remote data.
  - When using date-differentiated files on the System/36, the most current file is selected. When running from a System/36 to an iSeries server, the NRD entry must specify a member name of *LAST to access the last member of the database file, which is the file with the most recent date. If no member name is specified, *FIRST is used.
- The remote label in the NRD on a System/36 source must be in the syntax required by the target server.
- The number of records allocated for a file differs between the System/36 and the iSeries server. File space allocation on the System/36 is in block units (2560 bytes) while on the iSeries server, file space allocation is by number of records. For example, if a user asks for a sequential file capable of storing ten 100-byte records, the System/36 allocates 1 block of file space (2560 bytes), and uses as much of the file space as possible (2500 bytes), giving the user twenty-five 100-byte records.

  The iSeries server allocates exactly 10 records. If the user took advantage of this allocation method on the System/36, the file (nonextendable) created using DDM on the iSeries server might be too small.
- The DDM Change End-of-File (CHGEOF) command used on the System/36 is not supported on the iSeries server.

- The iSeries server does not support writing over data on the Delete File (DLTF) command. If an iSeries user accessing a System/36 wants to overwrite the data, an application program must be written on the iSeries server, or the user must access the target System/36 and perform the delete operation with the erase operation.
- A System/36 source server cannot create direct files on an iSeries target server that are nondelete-capable. The iSeries server does not support files that are nondelete-capable for all file organizations.
- The System/36 does not allow a record with hex FF in the first byte to be inserted into a delete-capable file. The iSeries server allows this.
- When running System/36 applications to another System/36, the application waits indefinitely for the resource to become available. When running System/36 applications to or from an iSeries server, these applications might result in time-outs while waiting for a resource to become available.
- Direct files are extendable on the System/36 but not on the iSeries server. If a direct file is created on the iSeries server as extendable, the file is allocated with three extents, but is never extended beyond the initial size plus three extents.
- The System/36 relay function is *not* supported whenever one of the servers in the network along the path from the source server to the target server is not a System/36. The iSeries server *never* supports the relay function; Advanced Peer-to-Peer Networking (APPN) must be used.
- Key fields on the System/36 are considered to be strictly character fields. The System/36 does not recognize a key field as being packed or zoned. Unexpected results might occur if source iSeries application programs refer to packed fields within a System/36 file. Because of the way packed numbers are stored and the fact that the System/36 does not recognize key fields as packed on relative keyed operations, records might be returned in a different order than expected or no record might be found when one is expected.

  As an example, the ILE RPG SETLL statement might fail unexpectedly with record not found or might select a record other than the record expected when using packed fields to a System/36 file. Only character and unsigned numeric fields should be used as key fields.

Related concepts

Migration

Related reference

"Rules for specifying target server file names for DDM" on page 112
The rules for specifying the name of a DDM file (on the local iSeries server) are the same as for any other file type on an iSeries server. The rules, however, for specifying the name of a remote file depend on the type of target server.

iSeries server and System/38 DDM differences

This topic consists of a list of differences between the iSeries server and System/38.

- Three parameters are added to the Create DDM File (CRTDDMF) and Change DDM File (CHGDDMF) commands. These parameters are the remote location name (RMTLOCNAME), local location name (LCLLOCNAME), and the remote network ID (RMTNETID). DDM files can be created either in the System/38 environment or on the iSeries server.
- The Submit Remote Command (SBMRMTCMD) command must be in the syntax of the target server, even in the System/38 environment. For example, when a System/38 submits commands to an iSeries server, the syntax of the iSeries server must be used.
- The remote file name must be in the syntax of the target server.
- The default value for the DDMACC parameter on the Change Network Attributes (CHGNETA) command on the System/38 is *REJECT. The default value for the DDMACC parameter on the iSeries server is *OBJAUT.
- On the System/38, files are created as FIFO (first in, first out) or LIFO (last in, first out). The default for creating a file is FIFO on the System/38.

When running System/38 applications that are dependent on duplicate keys being FIFO or LIFO to an iSeries server, you should specify either FIFO or LIFO when creating your iSeries files because there is no default for iSeries files. This means the iSeries server looks for an available index path to share, which can be either FIFO or LIFO.

- Keyed files containing fields other than character (zoned or packed) created by using DDM on a remote System/38 might result in the fields defined as character fields. This might produce unexpected results when such a file is processed using relative keyed operations. Because the file is created with fields that are not packed, records might be returned in a different order than expected or no record can be found when one is expected.

  As an example, the ILE RPG SETLL statement might fail unexpectedly with record not found or might select a record other than the record expected when using packed fields to a System/38 file. Only character and unsigned numeric fields should be used as key fields for files that are created by using DDM on the remote System/38.
- To support adding a record by the relative record number operation, an ILE RPG program is required for DDM to do a READ CHAIN(RRN) operation followed by a WRITE operation. The file must be opened for read and update authorities, and the user must have read and update data authorities to the file.

  Format selector programs on adding a record by the relative record number operation are only supported on the iSeries server. Incompatibilities might arise for those users who have a format selector program for a logical file if they do direct file processing.

Related reference

"Rules for specifying target server file names for DDM" on page 112
The rules for specifying the name of a DDM file (on the local iSeries server) are the same as for any other file type on an iSeries server. The rules, however, for specifying the name of a remote file depend on the type of target server.

Related information for distributed data management

Listed here are the product manuals, Web sites, and information center topics that relate to the distributed data management topic. You can view or print any of the PDFs.

Not all of these manuals are referred to in this topic collection. You may need to use one or more of the following IBM iSeries manuals and topics while using this topic collection.

Communications:

- The APPC, APPN, and HPR topic provides the application programmer with information about the Advanced Peer-to-Peer Networking (APPN) support provided by the iSeries server. This topic provides information for configuring an APPN network and presents considerations to apply when using APPN. This is also a guide for programming and defining the communications environment for APPC communications.
- SNA Distribution Services provides the system operator or administrator with information about configuring a network for Systems Network Architecture distribution services (SNADS) and the Virtual Machine/Multiple Virtual Storage (VM/MVS) bridge.
- ICF Programming provides the application programmer with information needed to write application programs that use iSeries communications and the i5/OS intersystem communications function (i5/OS-ICF).
- Communications Management provides the system operator with communications work management information, error handling information, communications status information, and communications performance information.

- Communications Configuration provides the application programmer with information about configuring line, controller, and device descriptions to communicate within a network. Additional configuration considerations are discussed.
- Remote Work Station Support provides the system administrator or user with concepts, examples, and information about preparation and configuration for using the display station pass-through function. This guide also contains information about using 3270 remote attachment, the Distributed Host Command Facility (DHCF) network, and the X.21 short hold mode (SHM) network.
- OptiConnect provides information about installing, using, and managing communications using OptiConnect.

Languages:

- See the *RPG/400® User's Guide* manual on the V5R1 Supplemental Manuals Web site.
- See the *System/36-Compatible COBOL User's Guide and Reference* manual on the V5R1 Supplemental Manuals Web site.
- See the *System/38-Compatible COBOL User's Guide and Reference* manual on the V5R1 Supplemental Manuals Web site.

Programming:

- ILE Concepts describes, for the application programmer, the concepts and terminology of the Integrated Language Environment of the i5/OS operating system. It includes an overview of the ILE model; concepts of program creation, run time, and debugging; discussion of storage and condition management, and descriptions of calls and APIs.
- The Control language topic provides the application programmer with a description of the iSeries server control language (CL) and its commands.
- The Application programming interfaces topic provides information about how to create, use, and delete objects that help manage system performance, use spooling efficiently, and maintain database files efficiently. This topic also includes information about creating and maintaining the programs for system objects and retrieving i5/OS information by working with objects, database files, jobs, and spooling.

Distributed Data Management (DDM) Architecture:

- *Distributed Data Management Architecture: General Information*, GC21-9527
- *Distributed Data Management Architecture: Implementation Planner's Guide*, GC21-9528
- *Distributed Data Management Architecture: Implementation Programmer's Guide*, SC21-9529
- *Distributed Data Management Architecture: Reference*, SC21-9526

Code license and disclaimer information

IBM grants you a nonexclusive copyright license to use all programming code examples from which you can generate similar function tailored to your own specific needs.

| SUBJECT TO ANY STATUTORY WARRANTIES WHICH CANNOT BE EXCLUDED, IBM, ITS
| PROGRAM DEVELOPERS AND SUPPLIERS MAKE NO WARRANTIES OR CONDITIONS EITHER
| EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR
| CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND
| NON-INFRINGEMENT, REGARDING THE PROGRAM OR TECHNICAL SUPPORT, IF ANY.

| UNDER NO CIRCUMSTANCES IS IBM, ITS PROGRAM DEVELOPERS OR SUPPLIERS LIABLE FOR
| ANY OF THE FOLLOWING, EVEN IF INFORMED OF THEIR POSSIBILITY:

1. LOSS OF, OR DAMAGE TO, DATA;
2. DIRECT, SPECIAL, INCIDENTAL, OR INDIRECT DAMAGES, OR FOR ANY ECONOMIC CONSEQUENTIAL DAMAGES; OR
3. LOST PROFITS, BUSINESS, REVENUE, GOODWILL, OR ANTICIPATED SAVINGS.

SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF DIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, SO SOME OR ALL OF THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU.

Appendix. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other 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 grant you any license to these patents. You can send license inquiries, in writing, to:

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

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

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

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

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

IBM Corporation

Software Interoperability Coordinator, Department YBWA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

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

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, IBM License Agreement for Machine Code, 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 contains 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 contains 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.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming Interface Information

This Distributed data management publication documents intended Programming Interfaces that allow the customer to write programs to obtain the services of IBM i5/OS.

Trademarks

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

| Advanced Peer-to-Peer Networking
| CICS
| DB2 Universal Database
| Distributed Relational Database Architecture
| DRDA
| e(logo)server
| i5/OS
| IBM
| IBM (logo)
| Informix
| Integrated Language Environment
| iSeries
| OS/400
| RPG/400
| SQL/400
| System/36
| System/370
| System/38
| z/OS

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft 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.

| Linux is a trademark of Linus Torvalds 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, and service names may be trademarks or service marks of others.

Terms and conditions

Permissions for the use of these publications is granted subject to the following terms and conditions.

Personal Use: You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative works of these publications, or any portion thereof, without the express consent of IBM.

Commercial Use: You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED ″AS-IS″ AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Printed in USA