SIEMENS

SIMATIC PCS 7 OSx

Remote Data Transfer Manual

Order Number: 6ES7 6550XX058BD2
Manual Assembly Number: 2811154–0001
Original Edition

⚠ DANGER

DANGER indicates an imminently hazardous situation that, if not avoided, will result in death or serious injury.

DANGER is limited to the most extreme situations.

⚠ WARNING

WARNING indicates a potentially hazardous situation that, if not avoided, could result in death or serious injury, and/or property damage.

⚠ CAUTION

CAUTION used with a safety alert symbol indicates a potentially hazardous situation that, if not avoided, could result in minor or moderate injury.

CAUTION

CAUTION used without the safety alert symbol indicates a potentially hazardous situation that, if not avoided, could result in property damage.

NOTICE

NOTICE indicates a potential situation that, if not avoided, could result in an undesirable result or state.

MANUAL PUBLICATION HISTORY

SIMATIC PCS 7 OSx 4.1.2 Remote Data Transfer Manual
Order Manual Number: 6ES7 6550XX058BD2
Refer to this history in all correspondence and/or discussion about this manual.

Event	Date	Description
Original Issue	7/02	Original Issue (2811154-0001)

LIST OF EFFECTIVE PAGES

Pages	Description	Pages	Description
Cover/Copyright	Original		
History/Effective Pages	Original		
Trademarks	Original		
iii — xii	Original		
1-1 — 1-12	Original		
2-1 — 2-13	Original		
3-1 — 3-20	Original		
4-1 — 4-17	Original		
5-1 — 5-11	Original		
A-1 — A-19	Original		
B-1 — B-1	Original		
C-1 — C-6	Original		
D-1 — D-1	Original		
E-1 — E-1	Original		
Index-1 — Index-4	Original		
Registration	Original		

Trademarks

SIMATIC®, SINEC®, and STEP® are registered trademarks, and S5™ and S7™ are trademarks, of Siemens AG.

PCS™, APT™, Series 505™, and TISOFT™ are trademarks of Siemens Energy & Automation, Inc.

Adobe® and Acrobat® are registered trademarks of Adobe Systems, Inc.

@aGlance™ and Net OLE™ are trademarks of Axeda, Inc.

Epson® is a registered trademark of Seiko Epson Kabushiki Kaisha.

Excel™ is a trademark, and Windows® and MS-DOS® are registered trademarks, of Microsoft Corporation.

HP®, DeskJet®, LaserJet®, and PaintJet® are registered trademarks of Hewlett‑Packard Company.

IBM® is a registered trademark of International Business Machines Corporation.

Intel® is a registered trademark of Intel Corporation.

Internet® is a registered trademark of Internet, Inc.

Lantronix® is a registered trademark of Lantronix.

Linux® is a registered trademark of Linus Torvalds.

Lotus® and 1‑2‑3® are registered trademarks of Lotus Development Corporation.

Network Computing Devices® is a registered trademark of Network Computing Devices, Inc.

Oracle® is a registered trademark of Oracle Corporation.

PostScript® is a registered trademark of Adobe Systems, Inc.

Red Hat® is a registered trademark of Red Hat, Inc.

TI™ is a trademark of Texas Instruments, Inc.

Tektronix® is a registered trademark of Tektronix, Inc.

UNIX® is a registered trademark of X/Open Company, Ltd.

VMS® is a registered trademark of Compaq.

X Window System™ is a trademark, and Motif® is a registered trademark, of the Open Group.

XESS® is a licensed, registered trademark, and AIS® is a registered trademark of Applied Information Systems, Inc.

Other trademarks are the acknowledged property of their respective holders.

Contents

Preface ..... **ix**

Chapter 1 Overview of Remote Data Transfer **1-1**

1.1 What Is the Remote Data Transfer Function? **1-2**
Preliminary Considerations 1-3

1.2 What Hardware and Software Components Make Up the Interface? **1-4**

1.3 How Does the Remote Data Transfer Option Operate? **1-6**
Directory Hierarchy 1-6
Operation 1-7

1.4 What Do I Need to Know before I Begin? **1-8**

1.5 Installing the RDT Option **1-10**
Installing RDT 1-10
Configuring Oracle Networking 1-11

Chapter 2 Creating Data Transfer Files **2-1**

2.1 Overview of a Data Transfer Program **2-2**
Associating Data Transfer with Specific Events 2-2
Creating a Data Transfer Specification File 2-3
Grouping Transfers by Event 2-4
Sequence of Tasks for Creating a Data Transfer Program 2-6

2.2 Assigning Data Types **2-7**
OSx Data Types 2-7
Methods for Assigning Data Types 2-8

2.3 Initiating Data Transfers **2-9**
Planning the Data Transfer Program 2-9
Planning Events/Times That Trigger Data Transfers 2-10
Planning Transfers for Multiple-Station Systems 2-11

2.4 Creating a File of Data Transfer Instructions **2-12**
Creating a File on the System 2-12
Creating a File on Another Machine 2-13

Chapter 3 Using RDT Programming Language 3-1

3.1 Overview 3-2
Types of Instructions 3-2
Conventions Used in the Instruction Descriptions 3-3

3.2 Programming Hints 3-4
Moving Several Rows of Data at One Time 3-4
Defining Transfer Items 3-5
Modifying/Deleting a Table 3-5
Using Bit Tests in a Condition Clause 3-5
Checking Number of Logins to the Remote Database 3-6
Oracle Date Option for RDT 3-6
Event Triggering from Controller Values 3-7

3.3 Default Instructions 3-8
Defining Database Write Type 3-8
Defining the Date Format 3-9

3.4 Alias Instructions 3-10
Defining an Alias 3-10

3.5 Data Transfer Instructions 3-12
Overview 3-12

Chapter 4 Managing an RDT Program 4-1

4.1 Overview 4-2
Interactive Tasks 4-2
Primary Role Change 4-3

4.2 Compiling a Program 4-4
Description of the Compile Process 4-4
Checking for Compile Errors 4-7
Debugging Compile Errors 4-8

4.3 Modifying a Program Description and Options 4-9

4.4 Installing a Program 4-11

4.5 Starting and Halting a Program 4-12

4.6 Removing a Program 4-14

4.7 Deleting a Program 4-15

4.8 Updating the RDT Program List 4-16

4.9 Transferring RDT Programs between OSx Stations 4-17

Chapter 5 Debugging Data Transfer Programs 5-1

5.1 Troubleshooting Problems during Compile and Installation 5-2
Using the Data Transfer Log File 5-2
Detecting Problems during Compile or Installation 5-3
Testing the Data Transfer Program before Installation 5-4

5.2 Troubleshooting Runtime Problems 5-6
Synchronizing in the Operate State 5-6
Using the Error Log File to Find a Problem 5-6
Sample OSx Log File Entry 5-6
Loss of Communication with Remote Database 5-7
Lost Connection Alarming 5-8
File System Alarm Goes into High High Alarm 5-8

5.3 Tracing the Operation of a Data Transfer Program 5-10
Sample Trace Outputs 5-10

Appendix A Data Transfer Error Messages A-1

A.1 Time/Event Scheduling Error Messages A-2

A.2 Database Error Messages A-4

A.3 Data Type Error Messages A-5

A.4 Constant Value Error Messages A-6

A.5 User Variable Error Messages A-7

A.6 File I/O Error Messages A-8

A.7 General Error Messages A-10

A.8 Internal Error Messages A-11

A.9 Syntax Error Messages A-12

A.10 Data Transfer Runtime Error Messages A-14

A.11 Error Classes A-17

A.12 Disabling Error Text on OSx Station A-18
Changing Error Message Destinations A-18

Appendix B Using Attributes to Trigger Events B-1
Appendix C Quick Reference for Instruction Syntax C-1
Appendix D Decimal/Hex Number Conversion D-1
Appendix E Date Formats for Year 2000 and Beyond E-1
Index Index-1

List of Figures

1-1 OSx Remote Data Transfer Function 1-2
1-2 Remote Data Transfer Components 1-4
1-3 Directory Structure for Remote Data Transfer Option Files 1-6
1-4 Operation of the Remote Data Transfer Option 1-7

2-1 Events and Their Resulting Data Transfers 2-2
2-2 Contents of a Data Transfer Specification File 2-3
2-3 Grouping Transfers by Event 2-5
2-4 Recommended Order of Tasks 2-6

4-1 Displaying the RDT Program Administration Dialog Box 4-3
4-2 Stages of Compiling a Data Transfer Specification File 4-4
4-3 Compile RDT Program Dialog Box 4-6
4-4 Creating the Administrative Log File 4-7
4-5 Modifying an RDT Program 4-10
4-6 Installing an RDT Program 4-11
4-7 Halting an RDT Program 4-12
4-8 Removing an RDT Program 4-14
4-9 Deleting an RDT Program 4-15
4-10 Updating the RDT Program List 4-16

5-1 Administrative Log File 5-2
5-2 Creating the Administrative Log File 5-3

List of Tables

1-1 Descriptions of Remote Database Interface Components 1-5
1-2 Basic Terminology 1-9

2-1 OSx Data Types 2-7

A-1 Time/Event Scheduling Error Messages A-2
A-2 Database Error Messages A-4
A-3 Data Type Error Messages A-5
A-4 Constant Value Error Messages A-6
A-5 User Variable Error Messages A-7
A-6 File I/O Error Messages A-8
A-7 Linux Error Codes and Possible Causes A-9
A-8 General Error Messages A-10
A-9 Internal Error Messages A-11
A-10 Syntax Error Messages A-12
A-11 Data Transfer Runtime Error Messages A-14
A-12 Error Classes A-17

B-1 OSx No-Event Attributes B-1

Preface

Purpose of This Manual

The *SIMATIC PCS 7 OSx Remote Data Transfer Manual* describes the remote data transfer feature of OSx. This feature allows you to transmit data collected from the process by an OSx station to the remote computer for historical records and other purposes.

- Chapter 1 provides an introduction to the remote data transfer option.
- Chapter 2 shows how to design and develop the structure for your remote data transfer program.
- Chapter 3 provides programming hints and gives the syntax for the RDT instruction set.
- Chapter 4 describes how to manage RDT programs interactively from the RDT Program Administration dialog box.
- Chapter 5 explains how to troubleshoot your data transfers.

If You Need Help

The use of remote data transfer (RDT) programs requires an adequate knowledge of system-level safety for the process being controlled, as well as an adequate knowledge of the remote database being accessed and of the database commands required for accessing the data.

If you have difficulty with your system, or if you have any questions regarding override considerations or the use of safety interlocks, contact the Siemens Energy & Automation, Inc., Technical Services Group in the U.S.A. at 800-333-7421, or in other countries at (49)-91-895-7000.

⚠ WARNING

Remote data transfer programs can affect the performance and functionality of your overall process.

Failure to observe required safety practices may result in unpredictable operation, which could cause serious injury or death to personnel, and/or property damage or loss of data.

You must verify that certain minimal safeguards, such as mechanical overrides and interlocks, are integrated into the overall process before attempting to use remote data transfer programs.

Conventions Used in the Manual Set

The procedures in the various manuals give you step-by-step instructions about how to carry out tasks. Typically, the last step of any procedure requires that you select the **OK** or **Save** button, press **Enter**, etc. To save space and reduce redundancy, this last step does not appear in the procedure. However, you need to finish each procedure with one of these actions.

OK Saves information that you have entered and closes the window.

Save Saves information that you have entered and does not close the window.

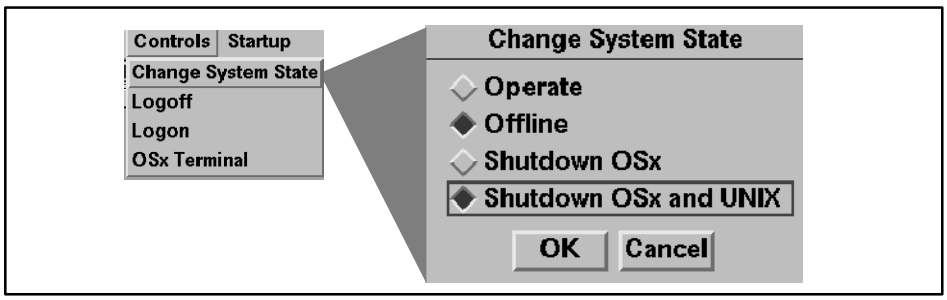
Cancel Closes the window without saving any information that you entered and terminates any action that you initiated.

Dismiss Closes the window without undoing any changes that you have entered. If you press **Dismiss** before you press **Enter**, the changes that you make are discarded.

The different fonts used in the manual set have the following meanings.

- Entries that you enter from the keyboard are indicated with the `courier font`.
- Items that you select on the screen, or keys that you press on the keyboard are indicated with this **bolded font.**

Items that you select on a cascaded menu are linked in the manual text with arrows. The first term indicates where to click the main menu bar. For example, **Controls->Change System State** tells you to click **Controls** on the main menu bar, then select **Change System State** from the pull-down menu.



Other Manuals

The OSx manual set consists of several manuals. If you cannot find the information that you need in the *SIMATIC PCS 7 OSx Remote Data Transfer Manual*, check these other books:

- *SIMATIC PCS 7 OSx System Administration Manual* This manual offers help in configuring network nodes and printers, and describes how to archive data and back up files.
- *SIMATIC PCS 7 OSx Process Configuration Manual* This manual describes the primary tasks required to configure your OSx system for controlling your process.
- *SIMATIC PCS 7 OSx Graphical Editor* This manual describes how to create the graphical displays used with OSx.
- *SIMATIC PCS 7 OSx Hardware Manual* This manual describes the hardware components of the system and how to install them.
- *SIMATIC PCS 7 OSx Reports Manual* This manual describes how to create reports on your process and your OSx configuration.
- *SIMATIC PCS 7 OSx Recipe Manual* This manual describes more advanced configuration tasks involving the creation and use of recipes.
- *SIMATIC PCS 7 OSx Batch Programming Manual* This manual describes more advanced configuration tasks involving the use of BCL, the Batch Control Language, and the creation of batch programs.
- *SIMATIC PCS OSx Operator Manual* This manual is written for the configuration engineer, but it describes how to carry out the various tasks that the process operator must do when the system is in the Operate state.
- *SIMATIC PCS OSx Interface to S5 Controllers Manual* This manual describes the OSx interface with controllers.
- *SIMATIC PCS 7 OSx Interface to S7 Controllers Manual* This manual describes the OSx interface with SIMATIC S7 controllers.
- *SIMATIC PCS 7 OSx Library Manual* This manual describes the function blocks used to program the S7-400 controllers to interface with OSx.

Be sure to check the Readme File for information that did not become available until after the publication deadlines for the PCS 7 OSx manuals. Select **Help->About OSx** from the main menu bar, and then click on the **Show Readme** button at the bottom of the About OSx dialog box.

Optional SIMATIC PCS 7 OSx Features

The following manuals are available for optional SIMATIC PCS 7 OSx features.

- *SIMATIC PCS 7 OSx X Terminal User Manual* This manual describes how to connect and operate an X terminal as an extension of an OSx station.
- *SIMATIC PCS 7 OSx @aGlance User Manual* This manual describes how to import OSx data into a Windows application, such as Excel or Lotus 1-2-3 or into another UNIX or VMS application.

Chapter 1

Overview of Remote Data Transfer

1.1 What Is the Remote Data Transfer Function? 1-2
Preliminary Considerations 1-3

1.2 What Hardware and Software Components Make Up the Interface? 1-4

1.3 How Does the Remote Data Transfer Option Operate? 1-6
Directory Hierarchy 1-6
Operation 1-7

1.4 What Do I Need to Know before I Begin? 1-8

1.5 Installing the RDT Option 1-10
Installing RDT 1-10
Configuring Oracle Networking 1-11

1.1 What Is the Remote Data Transfer Function?

The SIMATIC PCS 7 OSx remote data transfer function enables you to schedule the transfer of data between the OSx database and databases in remote systems (Figure 1-1). You schedule the data transfers between systems based on time events, OSx database events (changes), or combinations of the two.

Remote data transfers consist of user-definable programs that OSx automatically executes to transfer data between the OSx database and a commercially available relational database management system (RDBMS). The data transfers travel over a local area network (LAN).

OSx and the remote system can send or receive data. The remote system can use OSx-related data to generate reports, compile statistics, or initiate other activities. OSx can use the data from the remote system to change the operation of a supervised process or to generate reports.

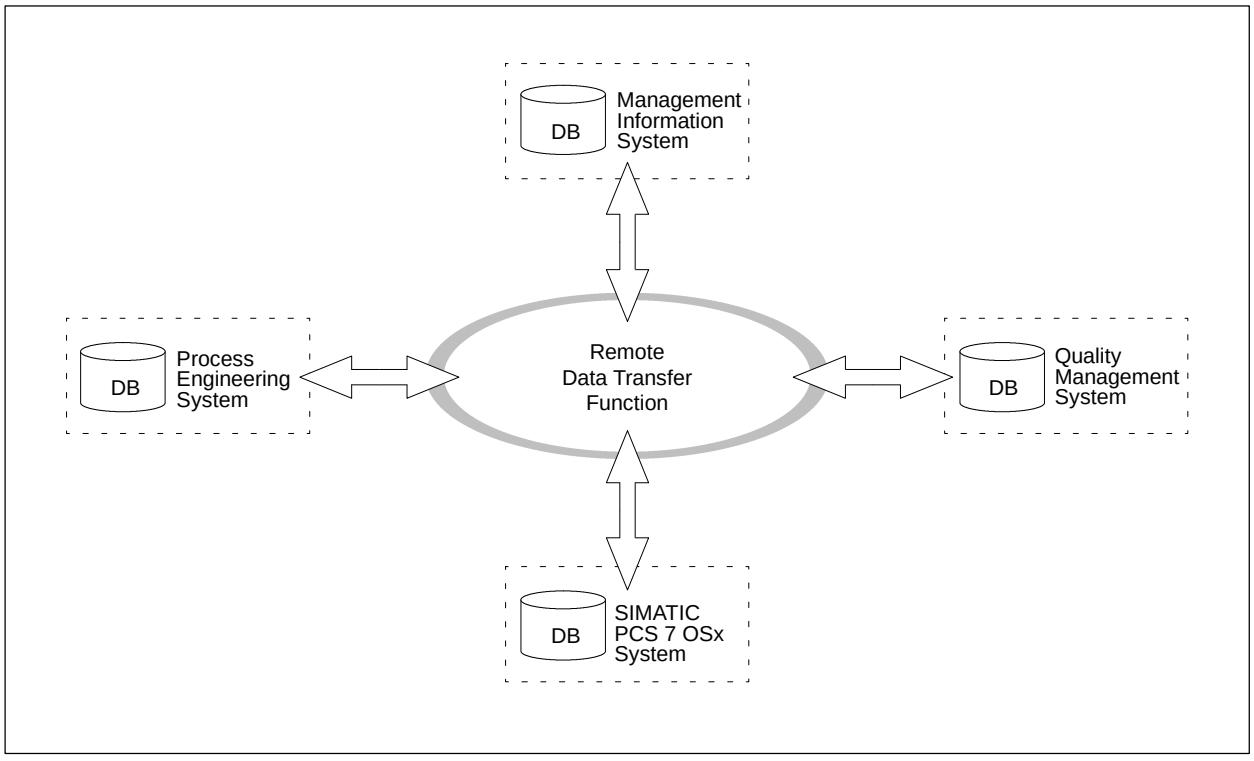


Figure 1-1 OSx Remote Data Transfer Function

Preliminary Considerations

Before you use the remote data transfer function, you need to make the following decisions.

- Determine which data items in the OSx and/or remote databases you want to transfer. You have the choice of either updating or appending data to the destination database.
- Determine which OSx database changes (events) require a data transfer. You also have the option to initiate a data transfer at a specific time of day or for a specific period of time.

After you determine these requirements, you are ready to define the data transfers. To define the data transfers and the events that drive them, you must create a data transfer specification file in which you enter the events and their corresponding data transfers using the instructions described in Section 2.4.

Before OSx can automatically execute the program, you must use the data transfer administration tool to compile and install the program so that it can run in the Operate state.

Refer to Chapter 4 for more information about administering data transfer programs. Refer to the *SIMATIC PCS 7 OSx System Administration Manual* for information about system states.

1.2 What Hardware and Software Components Make Up the Interface?

The remote data transfer function consists of both hardware and software components. The data transfer components and processes for both systems serve as the interface between the databases and the ethernet network (Figure 1-2). Some of the remote data transfer components are not normally supplied with SIMATIC PCS 7 OSx. Check with your remote system vendor for the hardware that is required to support ethernet TCP/IP communications on the remote system.

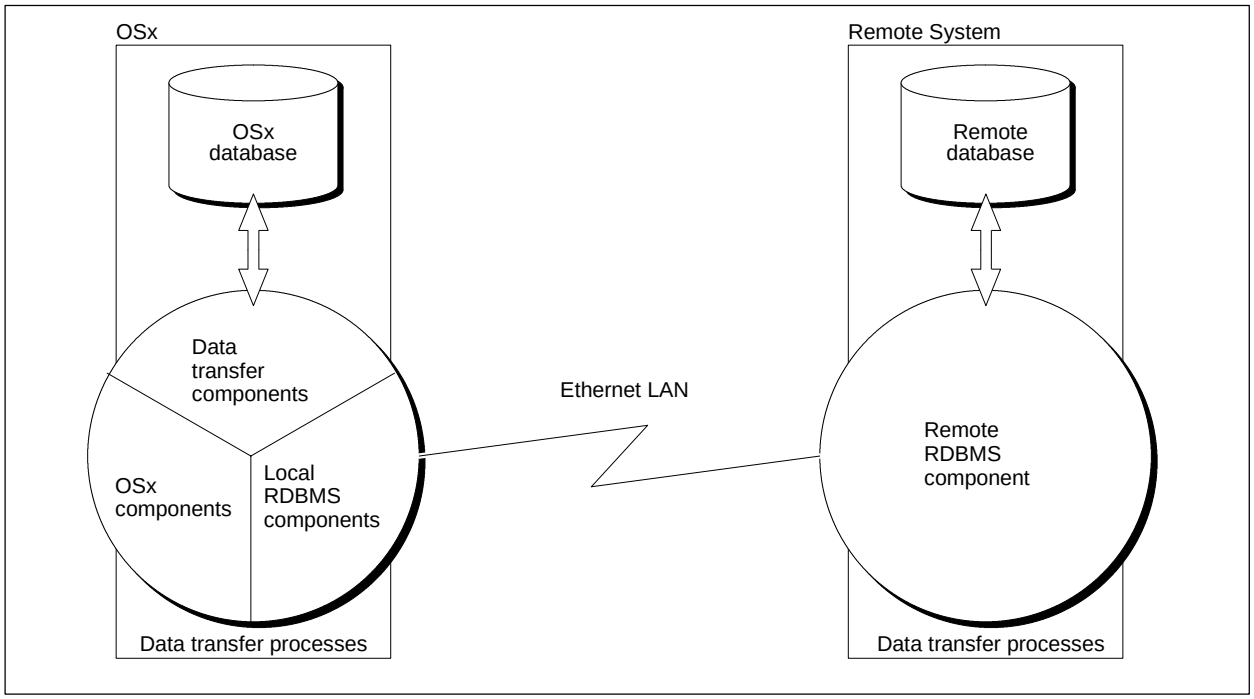


Figure 1-2 Remote Data Transfer Components

Table 1-1 describes each of the components that are illustrated in Figure 1-2.

Table 1-1 Descriptions of Remote Database Interface Components

Component	Description
OSx database	Contains data related to the tags that are supervised by OSx. Typically, the data in this database is stored and referenced by tag:attribute combinations.
Data transfer components	Consist of the following items: • Data transfer specification files — These files contain the data transfer instructions that you specify to move data from one database to another. You must associate each transfer with a specific OSx database change or time event. Refer to Chapter 2 for more information. • Data transfer programs — These programs are created when you compile and install data transfer specification files. OSx automatically runs these executable programs to transfer data between databases. Refer to Chapter 4 for more information. • Administration program — This program enables you to compile, install, and manage your data transfer programs. Refer to Chapter 4 for more information. • Several related directories — These directories are reserved for the trace output, compiler logs, intermediate files, and data queueing files. • Several intermediate source files — These files are created when you compile and install a data transfer specification file.
Local RDBMS components	Consist of a pre-compiler for SQL code (Pro*C/C++ for Oracle) and networking software (Net8 for Oracle).
OSx components	Consist of all the standard OSx programs, directories, and tables.
Ethernet LAN	Consists of the ethernet card and other network hardware/software for the remote system. You must purchase and install these components separately from the remote data transfer option.
Remote RDBMS component	Consists of a tool that runs on the remote system and serves as an interface between the RDBMS and the ethernet LAN. You must purchase and install this component separately from the remote data transfer option.
Remote database	Contains data related to the remote system (for example: engineering data or management statistics). Typically, the data in this database is stored and referenced by the relation name and attribute.

1.3 How Does the Remote Data Transfer Option Operate?

Directory Hierarchy

OSx stores the various files used by the remote data transfer option in several different directories. Some directories contain working programs; others contain temporary or intermediate files. Figure 1-3 illustrates the directory structure used by the remote data transfer option.

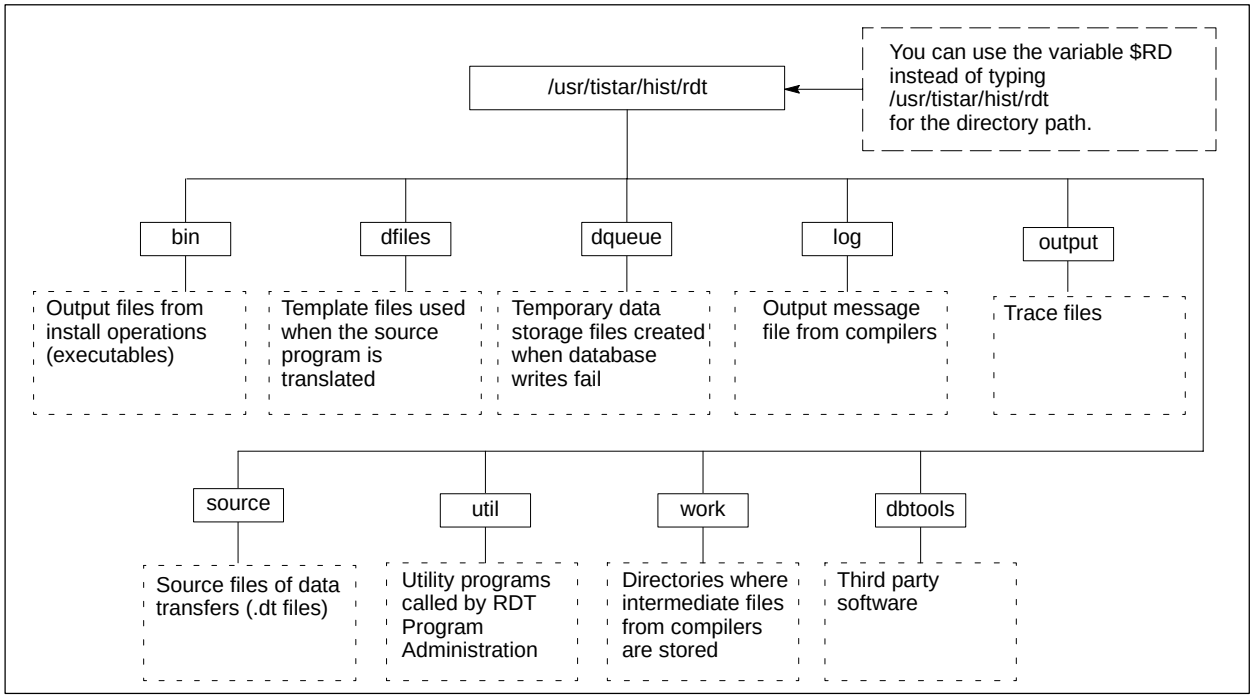


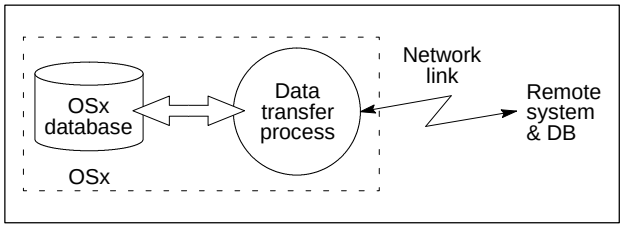
Figure 1-3 Directory Structure for Remote Data Transfer Option Files

Operation

After the transfer specification file is installed on OSx, the data transfers occur automatically when the system changes to the Operate state, according to the events described in the transfer program. Each data transfer program logs into the remote database as one user. Figure 1-4 shows how the transfer process operates.

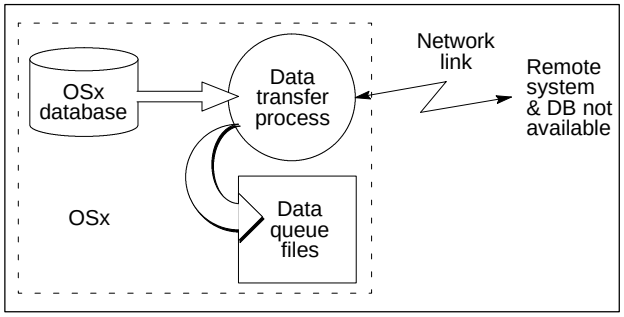
1 Normal operation

When a specific time or database event occurs, OSx executes the associated data transfer between the OSx database and the remote database.



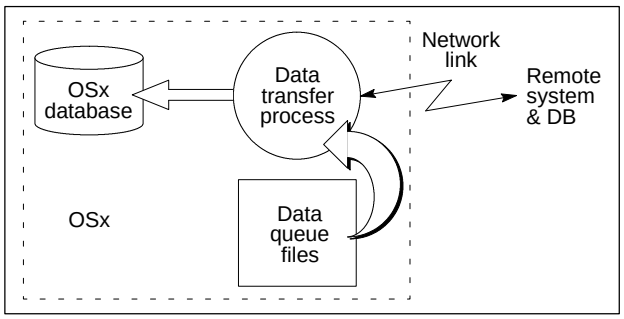
2 Operation when communications fail

If the network link fails or the remote system becomes unavailable, OSx stores the data in one or more files called data queue files.



3 Operation when communications are reestablished

When communications between OSx and the remote system are reestablished, OSx writes the stored data.



4 Operation after data queue files have been processed

After all of the data queue files have been processed, OSx deletes the files and resumes normal operation.

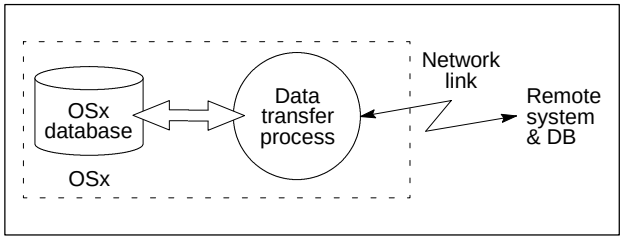


Figure 1-4 Operation of the Remote Data Transfer Option

1.4 What Do I Need to Know before I Begin?

Before you begin to configure and run the remote data transfer package, ensure that the following prerequisites are met.

- ❑ You have successfully installed the RDT software on all supervisory stations. (Section 1.5).
- ❑ You have purchased and successfully installed an ethernet LAN for your remote computer.
- ❑ You have purchased and successfully installed on your remote computer the appropriate networking software (SQL*Net or Net8 for Oracle) for your RDBMS.
- ❑ You must be familiar with relational databases and must know the difference between database updates and appends. You must also be familiar with how OSx and the remote system(s) process database transactions.
- ❑ You have added the name of the remote database system to the OSx networking file (**/etc/hosts**) and configured each supervisory station for RDBMS networking.

  You need to modify the appropriate networking files on the remote database system to allow for communication with OSx over the LAN.
- ❑ You must be familiar with OSx and its database and must know which database tables exist and where OSx stores its data. In addition, you must be familiar with the data types for each column (attribute) in the tables.
- ❑ You must be familiar with the remote system and its database. You must know which database tables exist and where the system stores its data. In addition, you must be familiar with the data types for each column in the tables.
- ❑ You must be familiar with the operating system. If you intend to create a transfer specification file in OSx, you must be familiar with one of the operating system text editors (for example: vi or gvim).

- ❑ You must be familiar with C-language programming. C-language experience can help you debug the programs used by the remote database interface.
- ❑ You must be familiar with the terms described in Table 1-2.

Table 1-2 Basic Terminology

Term	Definition
Attribute	A specific column associated with a tag in the OSx database. An OSx attribute functions like a column in other databases.
Column	A field that represents a specific kind of data in a database. A typical database might, for example, have a column that represents yearly salaries for all employees.
Commit	A command that permanently incorporates transaction changes in the database.
Data transfer	A transaction that moves data from one database to another.
Data transfer group	A group of one or more data transfers that are triggered by the same event or condition and that generate a transaction.
Data transfer process/program	An executable program that consists of one or more data transfer groups and their schedules. This process/program also contains machine definitions.
Relation	A group of data in the OSx database that is equivalent to a table.
Rollback	A command that removes transaction changes from the database.
Row	A field that represents one set of related data. For example, a database might have a row of data related to a single employee.
Table	A table consists of one or more rows and their related columns. A table is the basic data access structure in a relational database management system.
Transaction	A logical unit of work that consists of one or more database changes.

1.5 Installing the RDT Option

Installing RDT

If you ordered the RDT option with your SIMATIC PCS 7 OSx system, it is already installed for you. If you ordered the RDT option at a later date, follow the steps below to install it.

1. If OSx is running, select **Controls->Logon** on the menu bar and log in as `root`.
2. Click **Controls->Change System State** and select **Shutdown OSx**.
3. Press the left mouse button to bring up a root menu, select **New Xterm**, and log in as `root`.
4. Insert the CD with the title SIMATIC PCS 7 OSx V4.1.2 Disk 2 in the CD-ROM drive.
5. Type `install.new` and press **Enter**.
6. The screen prompts you for the SIMATIC PCS 7 OSx Options Activation Key. Enter the key exactly as it appears on the options license. A list of options appears that includes all options that are currently licensed for the OSx station. For example:

Number	Installed?	Licensed?	Description
1	Yes	Yes	E4.1.2 Supervisor S/W
2	No	No	E4.1.2 HP X Terminal S/W
3	Yes	Yes	E4.1.2 Tektronix X Terminal S/W
4	No	No	E4.1.2 RDT Oracle S/W
5	No	No	E4.1.2 RDT Sybase S/W
6	No	No	E4.1.2 One User OSx @aGlance Server S/W
7	Yes	Yes	E4.1.2 Ten User OSx @aGlance Server S/W

   If this is a supervisory station, or if other options have been previously installed, the above list appears first, and you are then prompted to enter the new SIMATIC PCS 7 OSx Options Activation Key. Your new key includes all previous options licensed for this station as well as RDT. Installation of RDT begins and may take several minutes.
7. When the installation is complete, the following prompt appears:

   Enter the number of an option to install/re-install, or enter a new OSx License String or enter q to quit.

   Type `q` to quit and press **Enter**.
8. Reboot the system. The RDT option is now installed on the station. When the OSx screen appears, remove the CD from the CD-ROM drive.

Configuring Oracle Networking

To configure Oracle networking, follow the steps below. Note that this version of RDT supports only the TCP/IP network protocol.

1. Consult the Database Administrator (DBA) for the remote database(s) and collect the following information:

   - The host name and IP address of the remote database system.
   - The port number of the Oracle "listener" process on the remote system.
   - The version of the remote Oracle database. The network interface supported by the OSx RDT package can communicate with two release levels of Oracle database products: Oracle8i and Oracle7. Oracle8 databases prior to release 8.1 are treated the same as Oracle7 in terms of the network interface.
   - The Service Name (for Oracle8i) or System Identifier (SID, for Oracle7 and Oracle8 prior to release 8.1) for the remote database.

2. Add the host name for the remote Oracle system and its IP address to the **/etc/hosts** file. An example is shown below:

   ```
   127.0.0.1      LOCALHOST
   209.109.10.150 OSx1
   209.109.10.99  ODBSRV
   ```

   In the example above, `OSx1` is the name of the OSx station. `ODBSRV` is the name of the remote Oracle system.

3. Open an Xterm window and log in as `tistar`.

4. Run the Oracle network configuration assistant by entering the following command: `netca`

5. On the Welcome window, choose **Local Net Service Name configuration**. On the next window, choose **Add**.

This procedure is continued on the next page.

6. Follow the instructions on the subsequent windows. Use the information collected in step 1 to fill in the configuration fields. Note the following special choices:

   - When you are asked to select a communications protocol, be sure to choose TCP.

   - At some point, you will be offered an opportunity to test the connection with the remote system. This test is optional.

7. The final piece of information for either release level of remote database is the selection of the Net Service Name. This is the name that is used to refer to the remote database within the RDT programs running on the OSx station. Since this name is significant only on the OSx station, you are free to choose whatever name you wish.

8. When you reach the end of the configuration sequence for the Net Service Name, you are asked whether you wish to configure another one. Be sure to press the **Next** button on this and subsequent windows until you return to the Welcome window. At this point, the **Finish** button will appear. You must press the **Finish** button in order to commit the information for the Net Service Name that you just configured.

Chapter 2

Creating Data Transfer Files

2.1 Overview of a Data Transfer Program 2-2
Associating Data Transfer with Specific Events 2-2
Creating a Data Transfer Specification File 2-3
Grouping Transfers by Event 2-4
Sequence of Tasks for Creating a Data Transfer Program 2-6

2.2 Assigning Data Types 2-7
OSx Data Types 2-7
Methods for Assigning Data Types 2-8

2.3 Initiating Data Transfers 2-9
Planning the Data Transfer Program 2-9
Planning Events/Times That Trigger Data Transfers 2-10
Planning Transfers for Multiple-Station Systems 2-11

2.4 Creating a File of Data Transfer Instructions 2-12
Creating a File on the System 2-12
Creating a File on Another Machine 2-13

2.1 Overview of a Data Transfer Program

Associating Data Transfer with Specific Events

With the remote database interface, you can write a program that transfers data when a specific database event or time event occurs. For example, you can designate that the current process variable for a boiler be uploaded to a remote system when a particular attribute changes, when 10:00 AM occurs, or when the second shift begins (Figure 2-1).

OSx also allows you to determine the "life cycle" of a data transfer. You can specify one or more of the following life cycle events for a data transfer transaction.

- A start event designates when or how a data transfer begins.
- A repeat event designates whether a data transfer occurs repeatedly after it starts. You can specify that a data transfer repeats when an OSx database event occurs, or that it repeats at a certain time interval.
- An end event designates when or how a data transfer schedule terminates.

The instructions for transferring data are stored in a data transfer specification file.

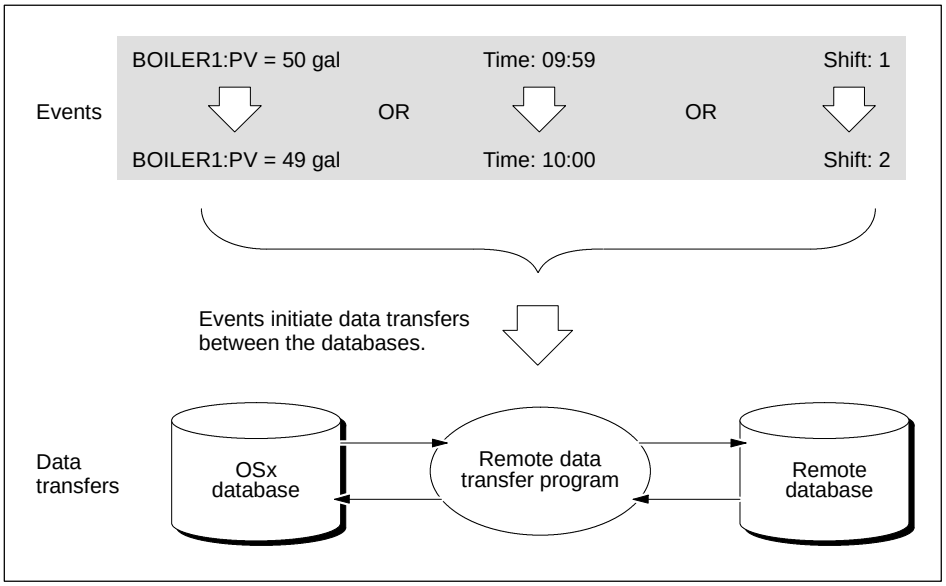


Figure 2-1 Events and Their Resulting Data Transfers

Creating a Data Transfer Specification File

A data transfer specification file contains the following instructions, as shown in Figure 2-2:

- Default instructions — A default condition statement remains true for each instruction that follows it until a subsequent default condition statement changes the original condition. If you want the default conditions to apply to the entire file, you must define the conditions at the top of the file.
- Alias instructions — An alias enables you to refer to remote machines and files from the data transfer statements. The alias definitions must appear before the transactions that use them; otherwise, OSx cannot recognize the alias names when they appear in the transactions.
- Data transfer instructions — Transfer instructions tell how and when to transfer data items.

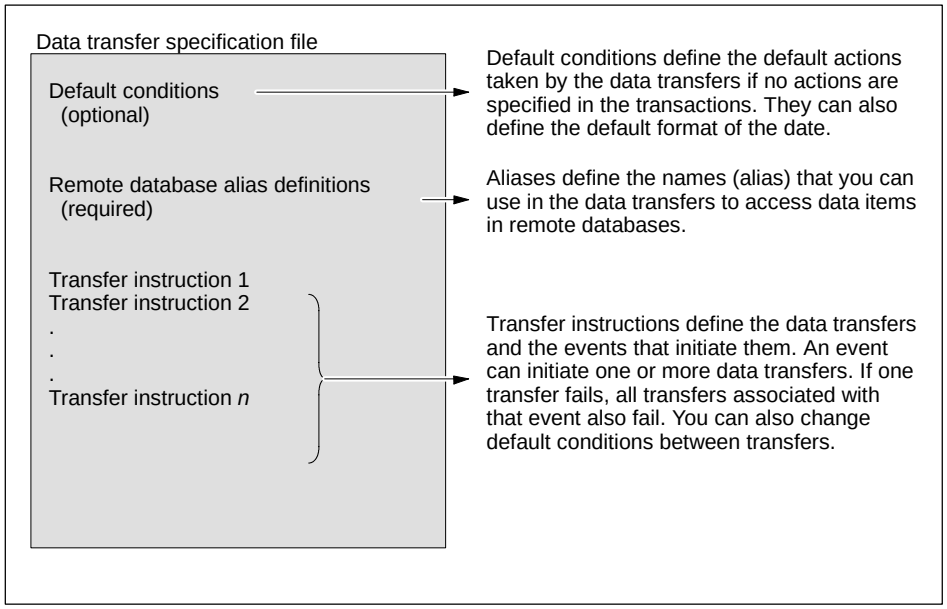


Figure 2-2 Contents of a Data Transfer Specification File

Grouping Transfers by Event

In a data transfer specification file, you group instructions that transfer data between databases when specific events occur. You can use the same event for more than one transfer group.

A transfer group consists of an event schedule and one or more transfer instructions enclosed in parentheses. OSx treats each transfer group as a single transaction; if any transfer within a group fails to read or write data, then none of the transfers in the group change data. In other words, OSx does not alter any of the database items unless all of the transfers within the transfer group are successful.

When you have multiple transfer groups within an RDT program, be aware of how the transfer groups interact. All transfer groups within the same program operate synchronously. This means that each transfer group, once triggered, runs to completion before the program looks for another trigger event and runs another group. A transfer group that takes a long time to execute will delay the triggering of other transfer groups in the same program. If transfer groups do not access the same OSx or remote database tables, or write to the same tag locations, you can place the transfer groups in separate programs to avoid the delay.

When you run multiple RDT programs at the same time, transfer groups that are located in different programs operate asynchronously and may interact unpredictably if they access the same OSx or remote DBMS database tables or write to the same tag locations. To avoid these interactions, place transfer groups that access the same tables or control points in the same RDT program.

Figure 2-3 illustrates a typical data transfer specification file with transfer groups. The shaded area indicates a single block.

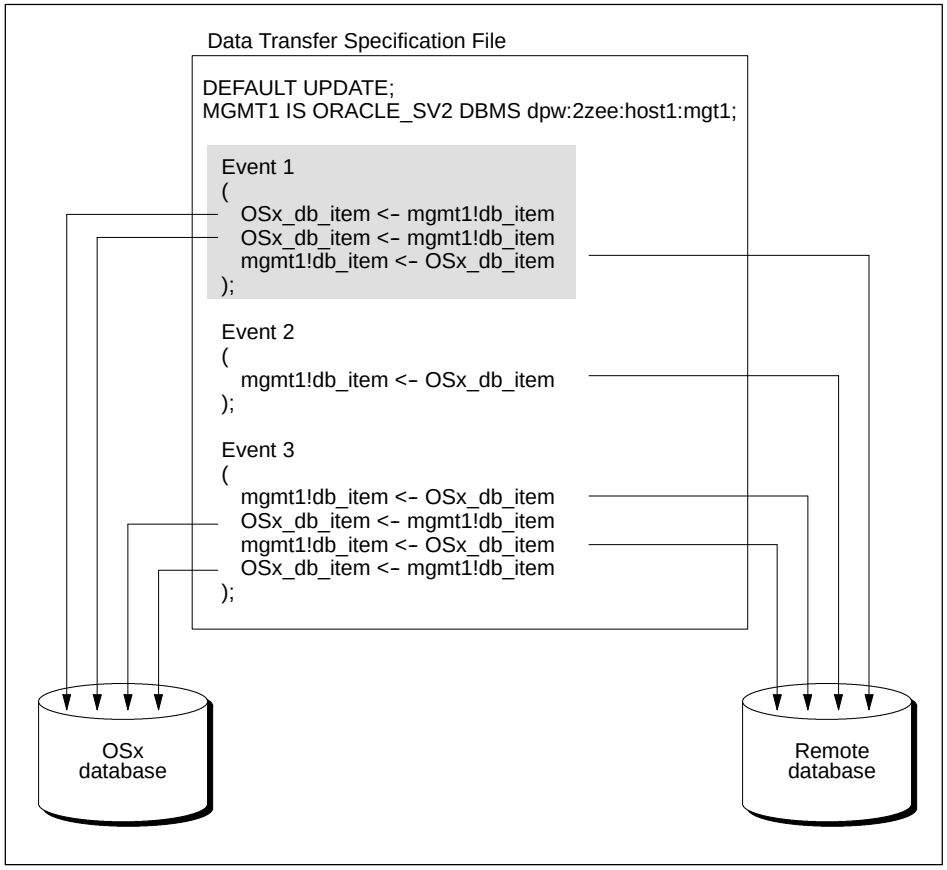


Figure 2-3 Grouping Transfers by Event

Sequence of Tasks for Creating a Data Transfer Program

Figure 2-4 lists the recommended order of tasks for configuring and installing a remote database interface.

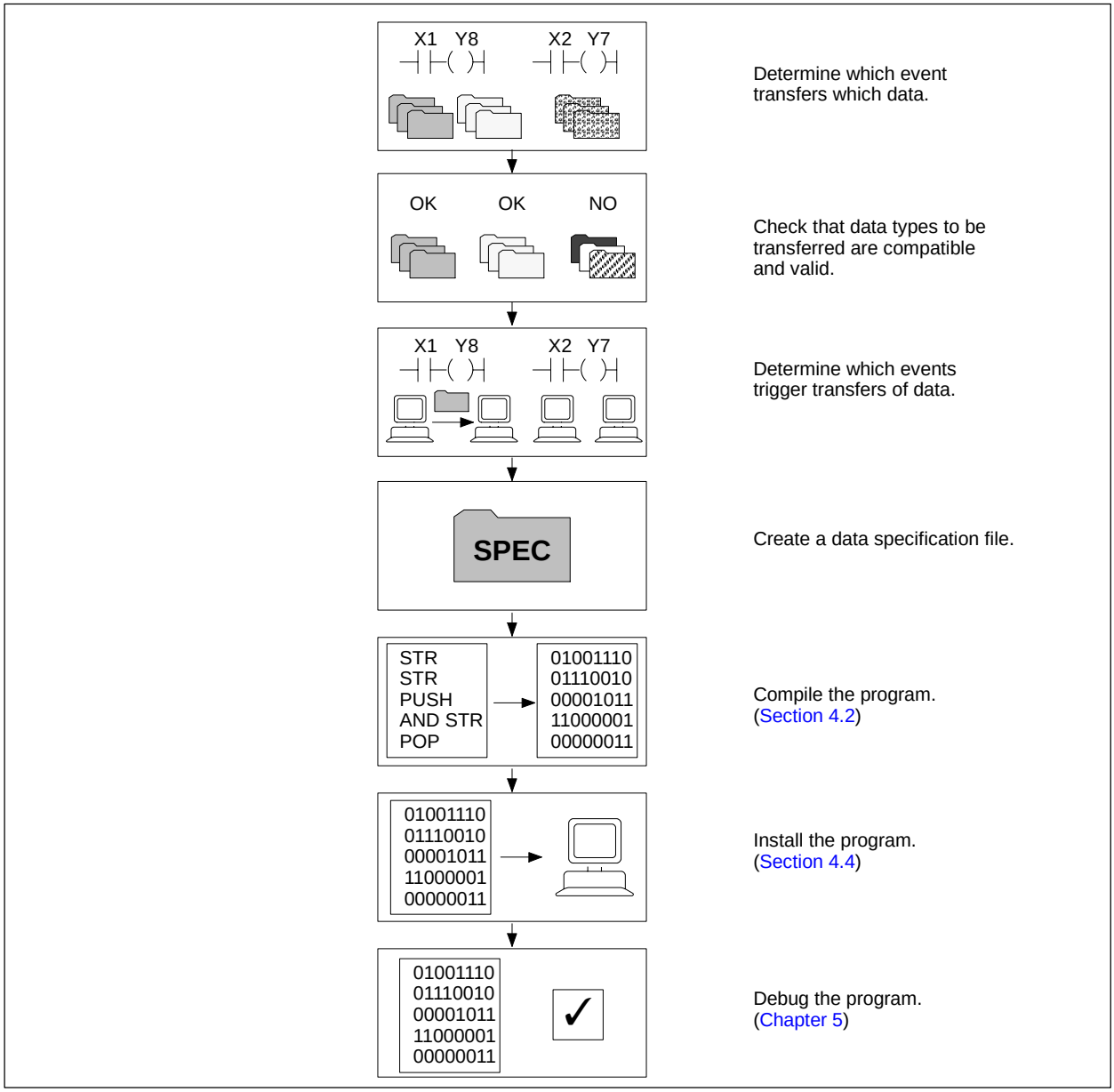


Figure 2-4 Recommended Order of Tasks

2.2 Assigning Data Types

OSx Data Types

All OSx database items have assigned data types. Where possible, the data transfer pre-compiler assigns a data type based on the data item's usage or appearance. If a remote database item does not have an implicit data type, you must assign one.

Table 2-1 shows the OSx data types and corresponding data transfer types. All other data items do not have explicit data types.

Table 2-1 OSx Data Types

OSx	Data Transfer Type	Valid Data Values
BYTE	SINT8	-128 to 127 or 0x80 to 0x7f
CISTRING	STRING	ASCII character string
DBBOOLEAN	SINT16	-32768 to 32767 or 0x8000 to 0x7fff
DBDOUBLE	FLOAT64	-1e+38 to 1e+38
DBLONG	SINT32	-2147483648 to 2147483647 or 0x80000000 to 0x7fffffff
DBSHORT	SINT16	-32768 to 32767 or 0x8000 to 0x7fff
LONG_STATUS	UINT32	0 to 4294967295 or 0x0 to 0xffffffff
LSET	UINT32	0 to 4294967295 or 0x0 to 0xffffffff
REAL_NUM	FLOAT32	-1e+38 to 1e+38
SCALAR	UINT16	0 to 65535 or 0x0 to 0xffff
SHORT_STATUS	UINT16	0 to 65535 or 0x0 to 0xffff
SSET	UINT16	0 to 65535 or 0x0 to 0xffff
STRING	STRING	ASCII character string
TIME_DATE	SINT32	-2147483648 to 2147483647 or 0x80000000 to 0x7fffffff
UBYTE	UINT8	0 to 255 or 0x00 to 0xff
ULONG	UINT32	0 to 4294967295 or 0x0 to 0xffffffff
UNION	N/A	This data type is not available. Use an appropriate data type for each value.
USHORT	UINT16	0 to 65535 or 0x0 to 0xffff

Methods for Assigning Data Types

OSx provides two methods for assigning data types:

- Implied data type assignment — If a data transfer program transfers a data item without a specified data type to or from the OSx database, the remote data transfer software assigns to the item the same data type as the corresponding OSx database item. In the following examples, OSx assigns the FLOAT32 data type to the user variable **%1** and the remote database item **present_value** so that they match the data type of the OSx PV attribute.

  ```
  %1 <- lp_0:pv
  lp_0:pv <-
  mgmt1!loop_points:present_value where (mgmt1!loop_points:key=5)
  ```

  The data type of a number constant is based on the format of the number. The following examples illustrate the implied data types for number constants.

25.3	**FLOAT32**
25	**SINT16**
0x40	**SINT16**

- Explicit data type assignment — If an OSx database item is not involved in a data transfer, you must explicitly specify a data type of a data item. It is also a good idea to directly assign data types to all remote database items to prevent the pre-compiler from assigning an invalid data type.

  To explicitly specify the data type for a number constant, user variable, and/or a remote database item, precede the data item with the data type enclosed in brackets. The following examples illustrate the use of explicit data types for data items.

  ```
  [FLOAT32]%1 <- lp_0:pv
  lp_0:out <-
  mgmt1!loop_points:[FLOAT32]output where (mgmt1!loop_points:key=5)
  LOOP_1:out <- [FLOAT32]23
  ```

2.3 Initiating Data Transfers

Planning the Data Transfer Program

Before you create a program of data transfer instructions (data transfer specification program), consider the following guidelines:

- ❑ Make a list of the data items that you want to read from a database and note the type of event that you want to initiate this read. Determine the destination of each data item that you plan to read from the database.

- ❑ Make a list of the data items that you want to write to a database, note the type of event that you want to initiate this write, and determine what type of write you require (i.e., update an existing row or append a new row). Determine the source of each data item that you plan to write to the database.

⚠ WARNING

Modifying a key attribute in one of the OSx database tables through an RDT program can lead to the loss of synchronization of databases between OSx stations.

Loss of synchronization can lead to unexpected controller operations, which can result in death or serious injury to personnel, and/or damage to equipment.

Do not use an RDT program to modify a key attribute in any of the database tables.

- ❑ While you can use any valid remote database user ID and password in a data transfer, adding a login for the **tistar** user ID with the password **tistar** (or whatever password you choose) allows you to specifically control access to the remote database by the system. You must give the **tistar** user ID the appropriate permissions in all of the remote database tables that you intend to read or write: if OSx attempts to access a table for which it does not have permission to access, the data transfer fails.

- ❑ For Oracle installations you need the server host name and the Net Service Name.

- ❑ Ensure that the data types of the source and destination database columns are compatible. Table 2-1 defines the data types in the OSx database. The data transfer pre-compiler assigns these data types to unknown items such as database columns and user variables. If you want to explicitly assign a data type to one or more of these items, refer to Section 2.2.

- ❑ The OSx database does not incorporate true NULL values. To eliminate unexpected results that may occur when a NULL value is transferred from the Oracle database to the OSx database, use the Oracle NOT NULL column constraint for those Oracle database columns that are read by a data transfer program.
- ❑ Review the data transfer instructions described in Section 3.1 and ensure that these instructions allow you to transfer data the way that you expect.

Planning Events/Times That Trigger Data Transfers

Keep the following considerations in mind when you plan the events and/or times for triggering data transfers.

- Ensure that the events are valid. If you try to use an invalid database relation for a trigger, the data transfer for that event will not occur.
- Determine whether a data transfer must occur when the current value of an OSx database item changes, regardless of the new value.
- Determine whether a data transfer must occur repeatedly at fixed intervals or at specific times during the day.
- Determine whether a data transfer schedule must terminate when either a database item changes or a specific time occurs.
- For transfers that have the same schedule, determine whether you want to group the data transfers in one transaction block, so that they execute when a single event occurs and fail if one of the transfers fail. If you want the data transfers to execute independently, give each transfer its own schedule and transaction block.
- Determine whether a data transfer must write a transfer “successful” or “unsuccessful” message to the OSx database when the transfer occurs. This task is required when the data transfer either updates or appends data to several rows in a database.

Planning Transfers for Multiple-Station Systems

If a failover occurs, then RDT programs have approximately ten seconds to complete the event block that they are executing. After this, the role change occurs and any database changes in the event block could be lost. Therefore, if failover has been implemented for the system, you need to design all event blocks to take less than ten seconds to execute. Check the trace files to see how long the event blocks take to execute, and be sure that all Oracle tables are indexed properly.

If a manual role change takes place, then RDT programs have approximately one minute to complete the event block that they are executing. After this, the role change fails if any event blocks are still executing. Therefore, to anticipate manual role changes, you need to design all event blocks to take less than one minute to execute. Check the trace files to see how long the event blocks take to execute, and be sure that all Oracle tables are indexed properly.

If a role change fails because an event block encounters an interruption in network communication with the remote database, the network timeout may exceed the role change timeout. Wait a few seconds and try the role change again. Be sure to resolve the network problem.

2.4 Creating a File of Data Transfer Instructions

Creating a File on the System

To create a file of data transfer instructions on an OSx supervisory station, follow these steps.

1. Open an XTerm and log in as `tistar`.
2. Type the following command at the prompt and press **Enter**. (Remember: you have the option of using the $RD variable—**$RD/source**—instead of typing the entire directory path.)

   cd /usr/tistar/hist/rdt/source

3. Open a file using one of the text editors (vi or gvim). The file name is the name of the data transfer program, and must be seven alphanumeric characters or less. The file name must include the **.dt** extension. If, for example, the file name is **prog1**, you must attach the **.dt** extension so that the file name is **prog1.dt**.
4. Enter the data transfer instructions.

   Chapter 3 describes the data transfer instructions available.
5. Save and close the file.
6. Proceed to Chapter 4 to compile the file and install the resulting data transfer program.

Creating a File on Another Machine

To create a file of data transfer instructions on a machine other than an OSx station, follow these steps.

1. Open an ASCII file using your favorite editor. The file name is the name of the data transfer program, and must be seven alphanumeric characters or less. The file name must include the **.dt** extension. If, for example, the file name is **prog1**, you must attach the **.dt** extension so that the file name is **prog1.dt**.

2. Enter the data transfer instructions.

   Chapter 3 describes the data transfer instructions available.

3. Save and close the file.

4. Copy the file to the **/usr/tistar/hist/rdt/source** directory, using either an MO disk or a file transfer software tool. You can do this on any supervisory station.

5. Proceed to Chapter 4 to compile the file and install the resulting data transfer program.

Chapter 3

Using RDT Programming Language

3.1 Overview 3-2
Types of Instructions 3-2
Conventions Used in the Instruction Descriptions 3-3

3.2 Programming Hints 3-4
Moving Several Rows of Data at One Time 3-4
Defining Transfer Items 3-5
Modifying/Deleting a Table 3-5
Using Bit Tests in a Condition Clause 3-5
Checking Number of Logins to the Remote Database 3-6
Oracle Date Option for RDT 3-6
Event Triggering from Controller Values 3-7

3.3 Default Instructions 3-8
Defining Database Write Type 3-8
Defining the Date Format 3-9

3.4 Alias Instructions 3-10
Defining an Alias 3-10

3.5 Data Transfer Instructions 3-12
Overview 3-12

3.1 Overview

Types of Instructions

A data transfer specification file contains these three types of instructions, defined in Chapter 2.

- Default instructions — A default condition statement remains true for each instruction that follows it until a subsequent default condition statement changes the original condition. If you want the default conditions to apply to the entire file, you must define the conditions at the top of the file.
- Alias instructions — An alias enables you to refer to remote machines and files from the data transfer statements. The alias definitions must appear before the transactions that use them; otherwise, OSx cannot recognize the alias names when they appear in the transactions.
- Data transfer instructions — Transfer instructions tell how and when to transfer data items.

Conventions Used in the Instruction Descriptions

The following pages describe the syntax and parameters for the instructions that you can use in your data transfer specification files. These descriptions use the following conventions.

- Text that you must type as shown appears in **bold**.
- Text that corresponds to variables or parameters appears in *italics*.
- The semicolon denotes the end of an instruction and must be used in those instances where it appears.
- Brackets [ ] are used to indicate optional parameters.

For further details about instruction syntax, see Appendix C.

3.2 Programming Hints

Moving Several Rows of Data at One Time

The remote data transfer language was designed to handle only one tuple (row) of data. In order to move multiple rows of data, each row must have a unique key value, and you must use additional control logic provided by either a custom program (using C or another programming language) or a batch control program.

1. Create two handshaking tags in OSx: one is used by the controlling program to trigger a data transfer, and the other is used by the data transfer to signal that the data retrieval was successful.

2. Create the data transfer program. Use a *start_event* (for example, DO_XFR) and a data retrieve status write tag (for example, GOT_DATA). See the following example:

```
on event DO_XFR:status changes
(
    %1<-xfr_tbl:key1
    %2<-xfr_tbl:key2
    log:val1,val2<-mytable:val1,val2 where (mytable:key1 = %1) and
     (mytable:key2 = %2)
)GOT_DATA:status = 1:0;
```

3. Create a controlling program, using either a batch control program or custom code (C or other language). This program must provide the following control logic:

    a. Initialize the status tag attribute (GOT_DATA in the example) to some value other than 0 or 1.

    b. Load the search key database item(s) with the key values for the search.

    c. Change the value of the transfer event tag (DO_XFR in the example) to start execution of the transfer.

    d. Check the value of the retrieve status attribute: if the value indicates success (in the example, value = 1) then one tuple (row) has been read successfully.

    e. Repeat steps b through d until the read status indicates failure. In the example, when there are no more rows of data to satisfy the search key, GOT_DATA:status = 0.

Each row of data to be retrieved must have a unique key value, regardless of whether the key consists of one or many attributes. If a table contains more than one row where each row's key attribute(s) contain the same value, then the same row will always be retrieved.

Defining Transfer Items

The Remote Data Transfer software does not check the following items during the compile process:

- Remote machine name — defined in **/etc/hosts** and in **/etc/hosts.equiv**
- Remote user ID — defined in the remote database.
- Remote user password — defined in the remote database.
- Remote database name — defined in the remote machine.
- Remote database type — defined in the remote machine.
- Remote database item names — defined in the remote database.

Specifying one of these items in a data transfer program without defining or assuring its existence generates an error at runtime. Ensure that each item has been defined, and check your data transfer program for spelling and typographical errors.

Modifying/Deleting a Table

Once a login accesses a table and as long as the login is active, the Oracle database does not allow that table to be deleted (DROP). The table itself is not locked — any user can change the data in the table — but the table cannot be deleted (DROP). To delete the table, you must terminate the data transfer process on OSx.

Using Bit Tests in a Condition Clause

The following statement tests a bit in a database item value:

```
relation:attribute:bit:bit_name [WHERE (relation:attribute = value)] = bit_value
```

The bit_value must be either 0 or 1 (off or on). Typically, only the status attributes of the tag relations have bit names assigned to individual bits. Refer to the chapter on configuring action requests in the *SIMATIC PCS 7 OSx Process Configuration Manual* for more information on bit names allowed.

NOTE: As with most OSx database item references, the WHERE clause is used only with a relation name, not if a tag name is specified.

Programming Hints (continued)

Checking Number of Logins to the Remote Database

The number of logins to the remote database may be limited. Since each OSx remote data transfer process uses a login, limit the number of separate transfer processes. Refer to the documentation for your database software.

For Oracle installations, be aware of potential problems with abnormal transfer terminations. The OSx data transfer process uses a two-task Oracle system when it remotely logs into the Oracle database. Some abnormal conditions (such as when the **Reset** button is pressed or when the LAN cable is disconnected during a data transfer) can leave the Oracle "shadow" process running and logged into the database. The presence of these shadow processes reduce the number of logins available. You must manually kill these shadow processes on the Oracle database machine to make logins available.

Oracle Date Option for RDT

When an RDT string timestamp statement (DT_TIME_S) is programmed as the data source in an RDT program, the default timestamp DD/MM/YY HH:MM:SS is generated and transferred to the remote database. If the Oracle date mode ora2 is activated, the string will be in the form DD-MON-YY. To activate the two-digit year Oracle date mode, enter the following command:

```
rdt_dtfmt ora2
```

If the Oracle date mode ora4 is activated, the string will be in the form DD-MON-YYYY. To activate the four-digit year Oracle date mode, enter the following command:

```
rdt_dtfmt ora4
```

To return to the default behavior, type `rdt_dtfmt std`.

Event Triggering from Controller Values

When you are using values that are updated by the controller, be aware that these values may change when OSx enters or leaves the Operate state. This can happen no matter what value is in the controller.

Example

For example, if you enter the following statement, every time an OSx state change occurs, the program is triggered.

```
ON EVENT Lp_0:status Changes
(    Transfer statement
);
```

If RDT programs use tag values for event triggers and the value in the controller changes faster than the Event Scan period, it is possible for different OSx stations to get different values for the same tag. If you have assigned a deadband to a tag, an RDT program running on one station may not get the same events as the same RDT program that is running on another station. If this happens, a role change involving the primary can lead to unexpected behavior by the RDT program.

⚠ WARNING

When tag values are used as event triggers, an RDT program that is running on one station may not get the same events as the same program that is running on another station.

This can lead to unexpected operation by the RDT program, which could result in death or serious injury to personnel and/or damage to equipment.

To avoid this, follow these guidelines:

- **Use a tag's status attribute to trigger the event. When this attribute changes, the change appears on all OSx stations.**
- **Set the Event Scan period (select Startup→Event Preferences on the menu bar) to be less than the change rate for all tag values used to trigger events.**
- **Set the deadband for all tag values used to trigger events to the lowest possible value.**

3.3 Default Instructions

Defining Database Write Type

In the data transfer specification file, you can define the default activity for data transfers by choosing one of several options for the default instruction. For example, you can designate that all data transfers are appended to the destination database. The default action applies to all transfers that follow it until you specify differently. To affect the operation of the entire data transfer specification file, this instruction must appear before all data transfer event groups.

Syntax

DEFAULT *update_type*;

Parameters

update_type a keyword that indicates whether the default action of a data transfer is an update or an append to a database. The valid values for this parameter are UPDATE, REPLACE, APPEND, and INSERT. If you do not specify a keyword, OSx uses the default keyword UPDATE.

The UPDATE and REPLACE options direct the data transfer to change an existing database value.

The APPEND and INSERT options direct the data transfer to add a new row to the specified database table.

NOTE: Be aware that setting the default action to APPEND or INSERT causes the database tables to grow with each data transfer.

Example

The following example sets the default action for a data transfer to UPDATE:

```
DEFAULT UPDATE;
```

Defining the Date Format

In the data transfer specification file, you can define the default format for the date that OSx uses to specify time/event dates.

Syntax

DEFAULT *date_format*;

Parameters

date_format Indicates which fields in the date are DAY, MONTH, and YEAR. If you specify the date format MONTH DAY YEAR and the event date is 04/03/92, then the assumed date is April 3, 1992. If you specify the format DAY MONTH YEAR and the event date is 04/03/92, then the assumed date is March 4, 1992.

When you use two digits to indicate the year, note that values from 92 to 99 are prefixed with “19” (for example, “93” becomes “1993”) and values from 00 to 37 are prefixed with “20” (e.g., “02” becomes “2002”). The values 38 to 70 are invalid.

The default format is DAY MONTH YEAR.

Example

The following example sets the default date format so that the date appears as year, month, and day.

```
DEFAULT YEAR MONTH DAY;
```

3.4 Alias Instructions

Defining an Alias

In the data transfer specification file, you must assign an alias to the combination of information required to uniquely identify a specific remote database. You then use this alias in the data transfer event groups to easily refer to and access the remote database. You must define an alias before you can use it in a transfer specification.

Syntax

alias **IS** *db_type* **DBMS** *user*:*password*:*id_1*:*id_2*;

Parameters

alias	Specifies the name to be associated with the remote database. Use this name to refer to the remote database in the data transfer event groups. This parameter is a character string of 15 characters or less.
db_type	Specifies the type of the remote database. The valid values for this parameter are ORACLE_NET8 and ORACLE_SV2. All data transfer programs use Oracle's Net8 protocol to connect to the Oracle remote computer. Net8 and SQL*Net V2 are compatible protocols. Use the ORACLE_NET8 and ORACLE_SV2 values to document the type of the remote database within the data transfer specification file.
user	Specifies the user ID that can access the remote database. This parameter is a character string.
password	Specifies the password for the user. The password must be valid for the name associated with the *user* parameter. The *password* parameter is a character string.
id_1	Specifies the hostname of the machine where the remote database resides. The name must correspond to a valid machine name listed in the **/etc/hosts** file. This parameter is a character string.
id_2	Specifies the Net Service Name for the remote database. The name must correspond to a valid database. This parameter is a character string.

Example The following example defines the alias **us35** for the user **doe1** whose password is **lq34t** on machine **mgmt1** and whose Oracle database name is **mdb1**. The example specifies a SQL*NET V2 connection.

```
us35 IS ORACLE_SV2 DBMS doe1:lq34t:mgmt1:mdb1;
```

NOTE: For every data transfer program running, there must be a database login.

3.5 Data Transfer Instructions

Overview

In the parameter descriptions, the parameter *db_item* refers to a specific database item in either the OSx database or the remote database. The *db_item* parameter has the following format.

relation:*attribute* **[WHERE (** *where_clause* **) [AND (** *where_clause* **) ...]]**

The *relation*:*attribute* parameters specify a particular instance in a database. For OSx tags, you can substitute the tag name for the *relation* parameter. For remote database items, you must precede the relation parameter with the remote machine's alias and an exclamation mark (!). For example, to access the **date** relation and **month** attribute on a remote machine with the alias **mgmt1**, enter the following values for the *db_item*.

```
mgmt1!date:month
```

The WHERE clause uniquely identifies a row in a relation. The *where_clause* parameter can consist of one data item specification or of several items joined by the logic operator AND. The database items within a WHERE clause can also contain WHERE clauses. Refer to the example at the end of this chapter for applications of WHERE clauses.

The WHERE clause is required in all cases except the following:

- A WHERE clause is not required if the *db_item* is an OSx tag name instead of a relation name.
- A WHERE clause is not required if the *db_item* is a transfer destination for an INSERT or APPEND action.

The following example transfer does not use a WHERE clause because the transfer destination is an APPEND to the database and, therefore, does not require a WHERE clause to identify a row.

```
mgmt1!status_log:status <- LP_0:status APPEND
```

The following example transfer requires a WHERE clause because the transfer destination is an UPDATE to the database. The transfer relies on the WHERE clause to identify the row in the database that is being updated.

```
mgmt1!loop:status WHERE (mgmt1!loop:tname='LP_0') <- LP_0:status UPDATE
```

Queries that reference controller-resident data as part of the key for selecting tuples from the database usually take longer than queries where all components of the key are resident in the OSx station. The extra time is required to allow updates from the controller to refresh the referenced data. The increase in access time is proportional to the number of controller-resident items in the WHERE clause and to the number of tuples that must be examined to satisfy the query. A single controller-resident datum (that is, a single attribute in a single tuple) may require up to 0.25 seconds for the refresh update.

Note that controller-resident data that is configured for RBE or armed for event notification does not require extra time since those data are maintained “fresh” by the controller communications subsystem.

Syntax

start_event [*repeat_event*] [*end_event*]
(
transfer_spec
) [*read_status*] ;

Parameters

start event Specifies the condition that initiates the data transfer(s) enclosed in parentheses. A maximum of 100 transfer groups are allowed per program. OSx executes the associated data transfer(s) when the start event occurs. If the event does not include repeat or end events, the data transfer occurs every time the start event is true. A start event can be a time or a change in a database item. Use one of the following formats for a start event.

ON EVENT *db_item* **CHANGES** [*condition_phrase*]
or
AT *time* [*condition_phrase*]

CAUTION

Specifying a start event of AT NOW causes the transfer group (and the transfer program) to run continuously.

This results in severe system performance degradation since it is always "now," and the transfer group is triggered continuously.

To avoid this situation, always qualify the NOW start event with a run phrase (REPEAT ON EVENT or REPEAT EVERY).

repeat event Specifies that data transfers must repeatedly take place after the start event transfer occurs. This is an optional event. A repeat event can be a time, an interval, or a change in an OSx database item. Use one of the following formats for a repeat event.

REPEAT ON EVENT *db_item* **CHANGES** [*condition_phrase*]
or
REPEAT EVERY *time_interval* [*condition_phrase*]

end event Specifies the condition on which the data transfer(s) terminate. This is an optional event. The default action for an end event is to stop the data transfer(s) forever regardless of the start event condition. The optional **RECYCLE** flag enables a start event to execute a data transfer again after an end event has triggered. Use one of the following formats for an end event.

UNTIL EVENT *db_item* **CHANGES** [*condition_phrase*] [**RECYCLE**]
or
UNTIL *time* [*condition_phrase*] [**RECYCLE**]

The symbol ▶ indicates a sub-parameter to a parameter listed in the syntax description.

▶*db_item* Corresponds to a specific database relation in either the OSx database or the remote database.

▶*time* Corresponds to a specific time. You can use 24-hour time or 12-hour time. If you use 12-hour time, you must indicate whether it is AM or PM. You can also specify a shift number (for example: SHIFT 1).

▶*time_interval* — Corresponds to a specific interval of time (for example: 2 HOURS). Time units are SECONDS, MINUTES, HOURS, DAYS, and WEEKS.

▶*condition_phrase* — Specifies an optional conditional event. The IF condition enables you to test the relationship between two items. You can test bits in database items only with the IF condition. The IF condition usually appears at the end of an event line, and it has the following format.

IF *condition_phrase*

The *condition_phrase* can either be a simple test of a single database item or be a complex logical statement that tests several database items. The *condition_phrase* typically has one of the following formats.

db_item relational_operator db_item

db_item relational_operator constant

condition_phrase logical_operator condition_phrase

The valid relational operators are

=	equal
>	greater than
<	less than
>=	greater than or equal
<=	less than or equal
<>	not equal

The valid logical operators are

AND	true if both conditions are true
OR	true if one or both conditions are true

The symbol ▶ indicates a sub-parameter to a parameter listed in the syntax description.

transfer_spec Specifies the database items and where to transfer them. A maximum of 50 transfer_specs are allowed per transfer group. A transfer specification can contain one or more data transfers. The general format of a transfer specification is:

data_destination **<-** *data_source* [*update_type*]

The *data_source* and *data_destination* parameters correspond to database items in either the OSx database or the remote database. The optional *update_type* parameter corresponds to one of the modification types: UPDATE, REPLACE, APPEND, or INSERT. (When the *update_type* is specified for a transfer, it overrides any **DEFAULT** *update_type*.)

▶*data_source* — Specifies the source of the data that you intend to transfer. You can use number constants, time stamps, OSx database items, and remote database items for the *data_source* parameter.

For database items used as transfer data sources, use the following format:

relation:*attribute1* [,*attribute2*,*attribute3*,...] [**WHERE (***where_clause***)**]

A maximum of 100 comma-separated attributes are allowed per transfer. Do not use the WHERE clause if the *relation* parameter is an OSx tag name. All other types of relations require the use of a WHERE clause. If you need to specify multiple attributes in a single data transfer, ensure that both the source and destination specifications contain the same number of items.

▶*time stamps* — Specify a time that is recorded when the transfer statement is first processed. The DT_TIME_N reserved word reads the current time as the number of seconds since January 1, 1970. The DT_TIME_S reserved word reads the current time as a string with the format “MM/DD/YY HH:MM:SS”. The actual position of the YEAR, MONTH, and DAY numbers is determined by the **DEFAULT** *date_format*. To change the Oracle date format for RDT, see page 3-6.

The symbol ▶ indicates a sub-parameter to a parameter listed in the syntax description.

▶*data_destination* — Specifies the destination of the data that you intend to transfer. You can specify user variables, OSx database items, and remote database items for the *data_destination* parameter.

⚠ WARNING

Modifying a key attribute in one of the OSx database tables through an RDT program can lead to the loss of synchronization of databases between OSx stations.

Loss of synchronization can cause unexpected process operation, which could result in death or serious injury to personnel and/or damage to equipment.

Do not use an RDT program to modify a key attribute in any of the database tables.

For database items used as transfer data destinations, use the following format:

relation:*attribute1* [,*attribute2*,*attribute3*,...] [**WHERE (***where_clause***)**]

Do not use the WHERE clause if the *relation* parameter is a OSx tag name or if the update is an APPEND or an INSERT. All other types of relations require the use of a WHERE clause. If you need to specify multiple attributes in a single data transfer, ensure that both the source and destination specifications contain the same number of items.

You can specify a user variable for the *where_clause* parameter. A user variable enables the data transfer specification to use a data item in one database in the WHERE clause for an item in the other database. You can also specify the user variable to read a single database item that can be used in several WHERE clauses.

Ten user variables are available, numbered 0 through 9. To load a user variable, designate it as the data destination for the transfer. You can then specify the variable in a data transfer's WHERE clause anywhere that you could use a data constant. The format for a user variable is a percent sign (%) followed by the variable number. OSx clears the user variables at the beginning of each transfer group. The following examples show you how to load a user variable (%0) and how to use it to download the setpoint for a tag.

```
%0 <- my_table:tag_name WHERE (my_table:current = 1)

my_table:sp WHERE (my_table:current = 1) <-
mgmt1!tag_data:set_point WHERE (mgmt1!tag_data:name = %0)
```

read_status Specifies the database item that the *transfer_spec* updates with the status of the data reads for a transfer group. The general format of the status is:

tag_db_item=*read_pass_value*:*read_fail_value*

The transfer specification updates the database item with the *read_pass_value* if all of the transfers in a group successfully read data. If any of the transfers fail to read the data, the transfer specification writes the *read_fail_value* to the database item.

For transfer groups that contain the *read_status* conditions, OSx does not generate the **Data Not Found** error message on the operator stations.

Example The following example shows a transfer group that runs every day between 7:00 a.m. and 6:00 p.m.; it copies a given loop tag's data into a remote loop information table, and then logs the loop tag's status into a remote log table. The transfers occur first at 7:00 a.m and then every time the item mytable:tag_ready changes AND the value of tag_ready is 1 or the status value is 0x40.

The first transfer loads user variable %1 with the name of the tag from mytable:tag_name.

The second transfer copies the named loop tag's status, process variable (pv), output value (out), and set point (sp) into corresponding columns in the remote database **loop_data** table for the row identified by tag_name.

The third transfer appends the named loop tag's status into the remote database table **loop_log**, which collects all status and setpoint values.

The fourth transfer updates the remote database item OSx_log:status with the current value of the OSx OSx1:status value. If all data was read successfully, rd_status:status is 1; otherwise, it is 0.

```
AT 7:00 AM today IF (mytable:OSx_stat = 0x20)
REPEAT ON EVENT mytable:tag_ready CHANGES
     IF ( (mytable:stat = 0x40) OR (mytable:tag_ready = 1) )
UNTIL 6:00 PM today
(
       [STRING]%1 <- mytable:tag_name

     mgmt1!loop_data:[UINT16]status, [FLOAT32]present_value,
               [FLOAT32]output, [FLOAT32]set_point
               WHERE (mgmt1!loop_data:tag_name = %1)
     <- loop:status, pv, out, sp  WHERE (loop:tag = %1)  UPDATE

     mgmt1!loop_log:stat, setp <- loop:status, sp  WHERE (loop:tag = %1)  APPEND

     mgmt1!OSx_log:status WHERE (mgmt1!OSx_log:OSx_name = 'OSx_1')
               <- OSx1:status
) rd_status:status = 1:0;
```

Chapter 4

Managing an RDT Program

4.1 Overview 4-2
Interactive Tasks 4-2
Primary Role Change 4-3

4.2 Compiling a Program 4-4
Description of the Compile Process 4-4
Checking for Compile Errors 4-7
Debugging Compile Errors 4-8

4.3 Modifying a Program Description and Options 4-9

4.4 Installing a Program 4-11

4.5 Starting and Halting a Program 4-12

4.6 Removing a Program 4-14

4.7 Deleting a Program 4-15

4.8 Updating the RDT Program List 4-16

4.9 Transferring RDT Programs between OSx Stations 4-17

4.1 Overview

Interactive Tasks

After you write an RDT program, you can manage it interactively from the RDT Program Administration dialog box. The following interactive tasks are supported.

- Compile the program (Section 4.2).
- Modify the program description (Section 4.3).
- Install the program (Section 4.4).
- Start and halt the program in the Operate state (Section 4.5).
- Remove (de-install) the program (Section 4.6).
- Delete the program (Section 4.7).
- Update the list of programs in the dialog box (Section 4.8).

You can display the RDT Program Administration dialog box on one or more supervisory stations simultaneously, but you cannot administer a particular program from more than one station at a time.

To display the RDT Program Administration dialog box, select **Data->RDT Program Administration** from the menu bar (Figure 4-1). The system displays all the RDT programs that you have created. All programs that you store in the **/usr/tistar/hist/rdt/source** directory appear in the dialog box.

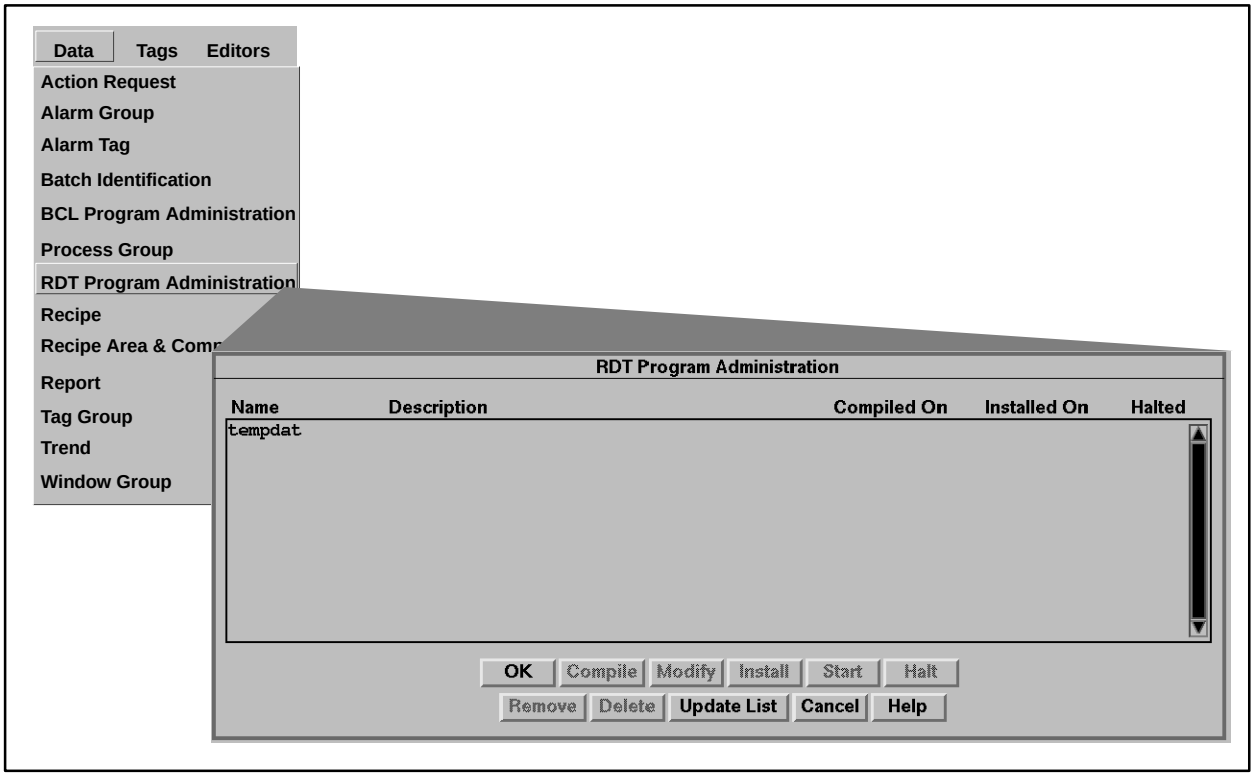


Figure 4-1 Displaying the RDT Program Administration Dialog Box

Primary Role Change

Do not use RDT Program Administration during a role change to a new primary. Normally, the OSx system prevents you from doing this; however, under heavily loaded conditions, you may be able to start RDT Program Administration before the system has propagated the start of the role change to all nodes.

4.2 Compiling a Program

Description of the Compile Process

Figure 4-2 shows the stages of the compile process. These intermediate files consist of C-language code, TQL code, SQL code, and object code, and they enable you to debug the program if any errors occur.

When you compile a program, OSx checks the data transfer and event specifications for syntax errors, verifies the existence of the OSx database items, and creates an executable file and several intermediate files. Any messages generated during compiling are stored in the administrative log file (**<*program_name*>.log**), located in the **/usr/tistar/hist/rdt/log** directory.

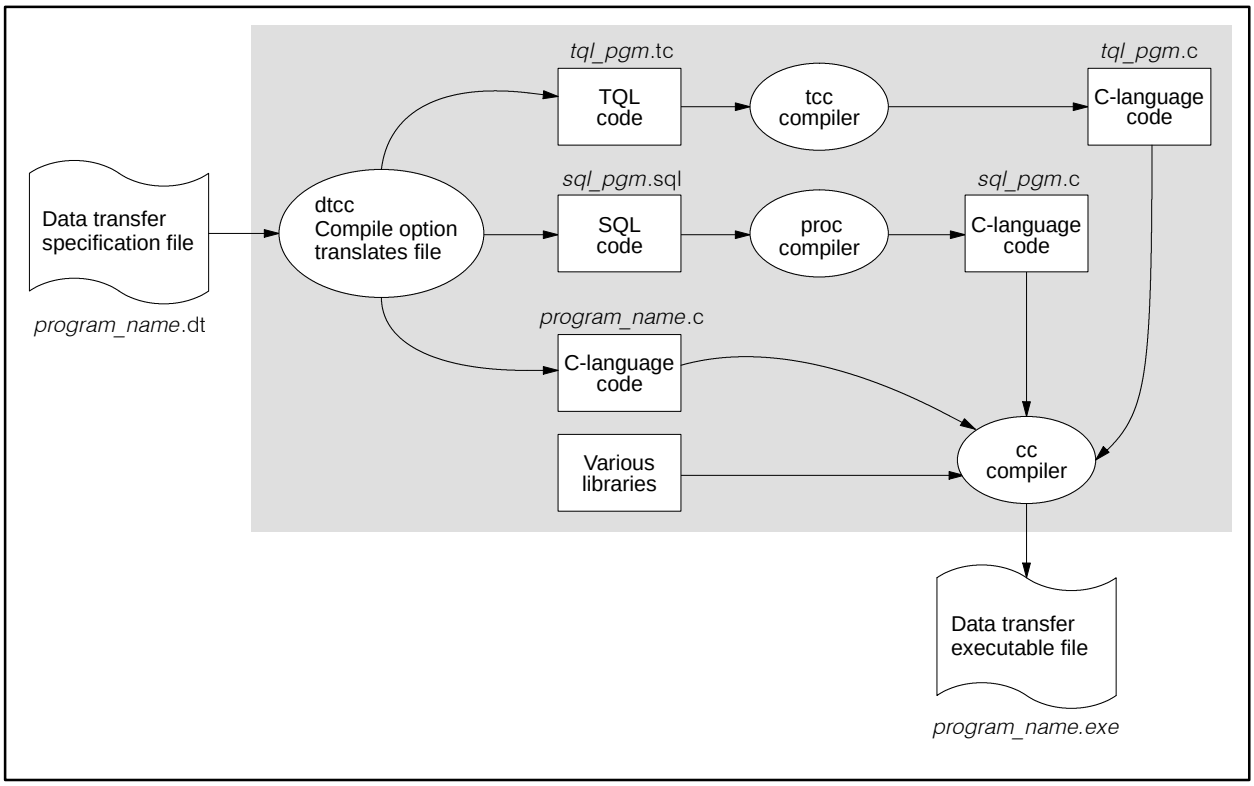


Figure 4-2 Stages of Compiling a Data Transfer Specification File

After you write an RDT program, you must compile the program before installing it. When the program is successfully compiled, the system copies the executable file to all supervisory stations. The path for the file is **/usr/tistar/hist/rdt/work/<pgm> .exe**, where **<pgm>** is the RDT program name without the **.dt** extension. When the compile is unsuccessful, the system displays the file containing the compiler errors and the "Compiled On" column shown in Figure 4-3 displays a string of asterisks (**********).

⚠ WARNING

If you alter or delete a tag, but do not recompile the RDT program, invalid data can result when the program is executed.

Invalid data can cause unpredictable controller operation, which can result in death or serious injury to personnel, and/or damage to equipment.

To avoid unpredictable controller operation, always recompile your program after you alter or delete tags. Tags must be deleted manually from the program; you cannot delete a tag from the OSx database until all references to it in the program have been removed and the program has been properly recompiled.

The system stores information about programs in the RDT_PGMS database table. You can compile up to eight RDT programs for execution.

Do not attempt to compile an RDT program that is currently running. If you need to compile an RDT program that is running, halt the program first. To determine if an RDT program is currently running, examine the Halted column in RDT Program Adminstration. An entry of **No** means the program is currently running.

To compile an RDT program, highlight the name of a program on the RDT Program Administration dialog box, and select the **Compile** button. The Compile RDT Program dialog box appears (Figure 4-3).

If you select **Yes** for the Debug option, or if any errors occur, the system saves all **.h** and **.c** (source code) files created during the compile. These files are located in the **/usr/tistar/hist/rdt/work/<pgm>** directory, where **<pgm>** is the RDT program name without the **.dt** extension. You can examine these files to troubleshoot your program for syntax and other kinds of errors. If you select **No** for the Debug option and the program compiles without error, the system saves none of these files, thereby conserving disk space.

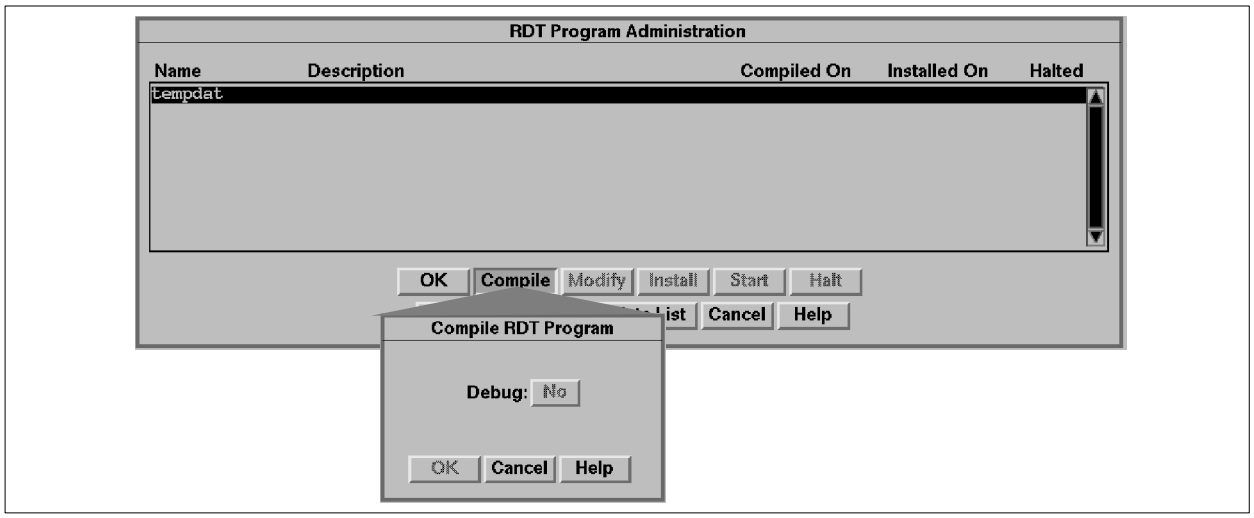


Figure 4-3 Compile RDT Program Dialog Box

Checking for Compile Errors

OSx creates a file of messages generated while the executable data transfer program is being created. The file **<*program_name*>.log** is located in the **/usr/tistar/hist/rdt/log** directory. Figure 4-4 shows the stages of compiling a data transfer specification file and the stages that write messages to the administrative log file.

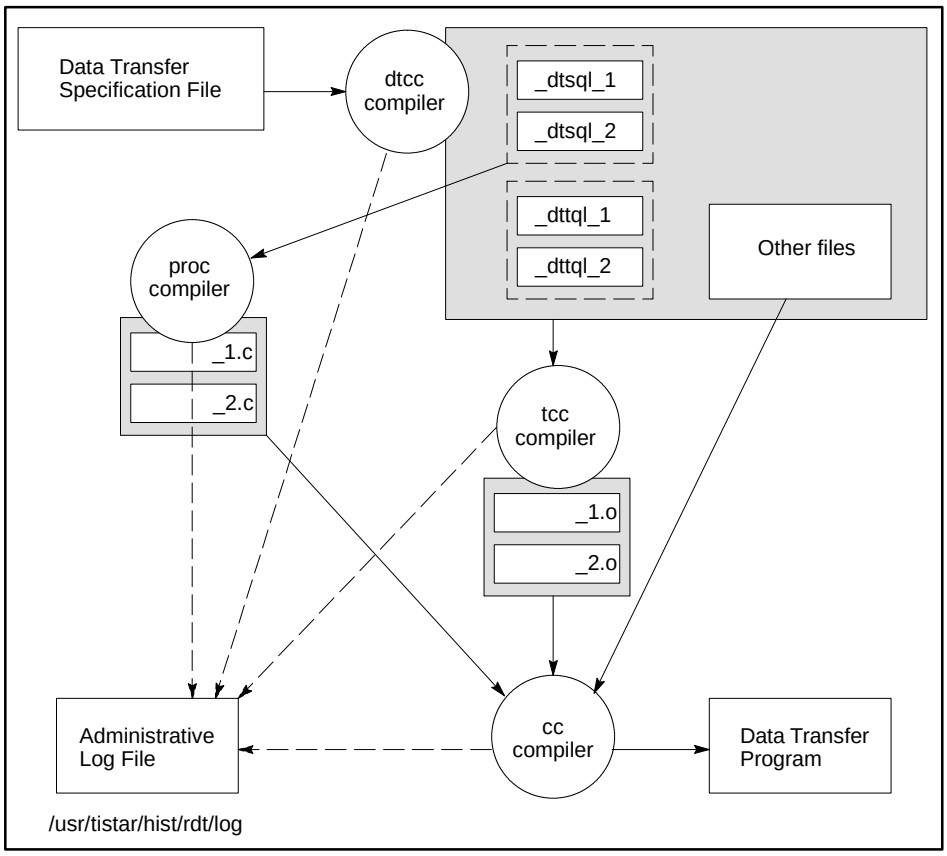


Figure 4-4 Creating the Administrative Log File

Debugging Compile Errors

If the compiler encounters any syntax errors, unmatched data types, or invalid data transfer specifications in the source program, it generates warning and error messages. You must correct all errors and compile again before you install the program.

Refer to Section 5.1 and Appendix A for a list of the possible data transfer warnings, errors, and their solutions.

NOTE: You cannot delete a tag from the system if it is currently used in an RDT program. Remove the tag from the program first, and then recompile and reinstall the program. Then you can delete the tag from the system.

4.3 Modifying a Program Description and Options

After you compile an RDT program, you can add or change its description and specify various runtime options. To modify an RDT program, click on the program name, and select the **Modify** button on the RDT Program Administration dialog box. The system displays the Modify RDT Program Information dialog box (Figure 4-5). Enter an appropriate description and select programming options from the following fields.

NOTE: If you select an RDT program from the RDT Program Administration dialog box after another user has already selected the same program on another station, the system does not allow you to modify any information about the program. The program remains selected and you can view any of the information related to compiling or modifying the program.

When the user on the other station finishes with the program, you still cannot modify the program until you deselect the program and then reselect it.

Trace Lines The number of lines entered for the trace output file determines how many of the last trace messages are stored. A good rule of thumb is five lines per transfer. The maximum number of lines allowed is 50,000. For more information on the trace function, refer to Section 5.2.

Set the size of the trace output file large enough to hold approximately 24 hours-worth of messages, yet small enough that the **usr/tistar/hist** file system does not become too full.

Connection Mode When you select **Continuous** for the connection mode, the RDT program logs into the remote system and remains logged in even when no data transfers are taking place.

When you select **On Demand** for the connection mode, the RDT program logs onto the remote system only when triggered by an event. The program logs off the remote system after the data transfer is complete.

The advantage of the Continuous mode is that data transfer is faster than the On Demand mode. The advantage of the On Demand mode is that loss of network communications with the remote system do not affect the execution of the RDT program. The database on the remote machine also limits the number of concurrent logins.

Connection Retries Enter the number of times that you will allow a connection failure before the system flags an error and logs a message in the **log.out** file.

If the RDT program fails to connect with the remote system in order to write data to the database, it retries at fixed intervals until it succeeds. If the program fails to connect in order to read data from the database, it makes no further attempts to connect.

Priority Choose the Linux priority at which you want the RDT program to execute. If you are unfamiliar with Linux process priority, use the Linux default priority, which is 20.

If you have already installed a program, you can still use the Modify feature to change the description or programming options. If the program is running when you modify it, you must halt the program and restart it for your changes to take effect.

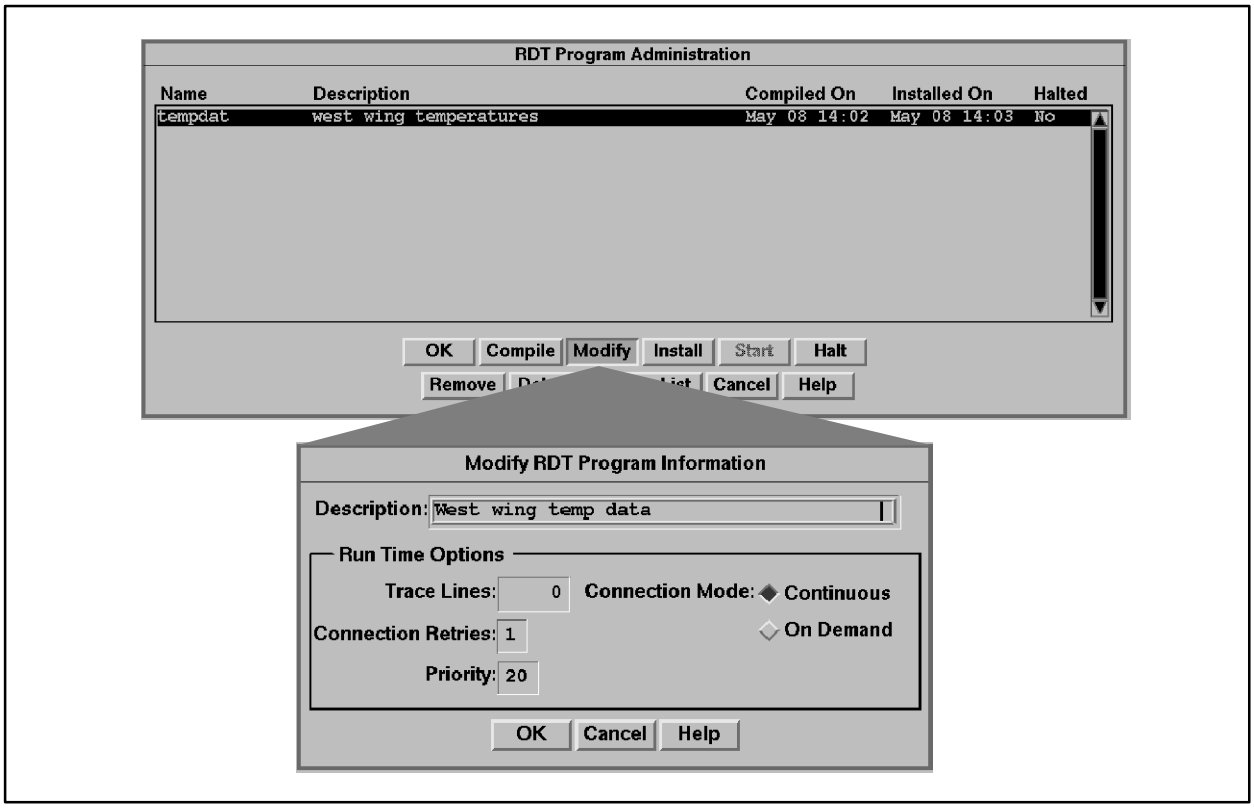


Figure 4-5 Modifying an RDT Program

4.4 Installing a Program

In order for a compiled program to execute automatically when the system transitions to the Operate state, you must install it. To install an RDT program, click on the program name, and select the **Install** button on the RDT Program Administration dialog box (Figure 4-6). The system prompts you to confirm your choice and then installs your program. OSx stores the RDT executable files in the **/usr/tistar/hist/rdt/bin** directory.

The number of data transfer programs that you can install is limited to no more than four programs to be run at any one time. If you need to reinstall a program and it is already running, you must halt the program first.

It is important to test programs before allowing them to operate with the factory process. See Chapter 5 for information about debugging RDT programs.

⚠ WARNING

An online data transfer program can alter the OSx database, thereby affecting values in the controller and/or causing the process to operate unexpectedly.

Unexpected operation of the controller can cause serious injury or death to personnel, and/or damage to equipment.

To prevent this risk, ensure that OSx is not interacting with the controller before you continue this procedure.

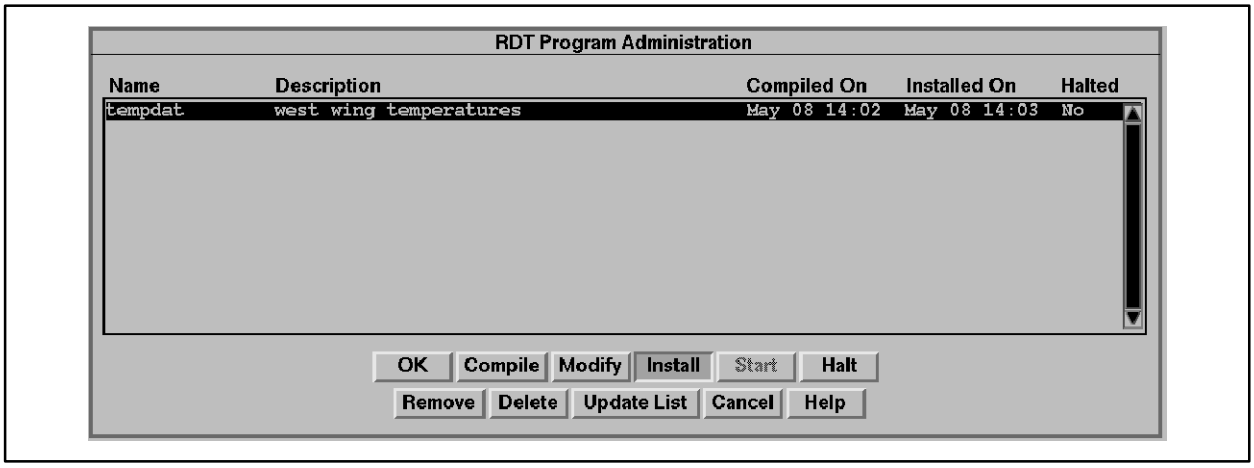


Figure 4-6 Installing an RDT Program

4.5 Starting and Halting a Program

After the RDT program is compiled and installed, it is available for execution. When OSx transitions from the Offline state to the Operate state, the system executes installed RDT programs using the program options that you have chosen. The programs continue running until OSx transitions to the Offline state. A transition back to the Operate state restarts the programs.

To halt an RDT program, click on the program name, and select the **Halt** button on the RDT Program Administration dialog box (Figure 4-7). The system prompts you to confirm your choice. After you halt a program, the system enables the **Start** button and the Halted column displays **Yes**. You can select this button to restart the program, and the Halted column then displays **No**.

If you halt a program, it does not start again until you restart it by selecting the **Start** button. When OSx transitions from Offline to Operate, no program that you halt manually resumes execution until you restart it manually.

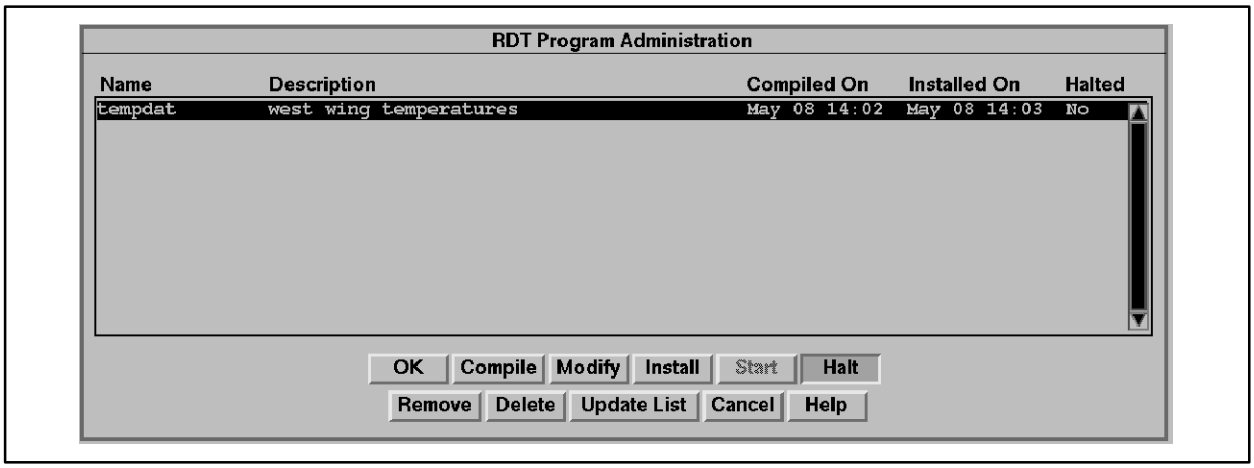


Figure 4-7 Halting an RDT Program

If an RDT program executes an infinite loop, it can never return to an idle state where it is looking for program events. A program in such a condition cannot process a Halt command. If you select the **Halt** button in the RDT Program Administration dialog box, the system displays **Yes** for halted, even though you cannot actually halt the program.

If you suspect that a program is running when the system shows that it is halted, you can kill it using operating system commands. Follow the steps below.

1. Open an Xterm window and log in as `tistar`.
2. Enter the following command: `ps -ef | grep <program name>`
3. Note the first number that appears on the line containing your program executable (not the grep command), if it exists. This is the process id (pid) for the program. If it does not appear, your program is already halted. If it is still running, go to step 4.
4. Kill the program by entering the following command:

   `kill <pid>`

   where `<pid>` is the process id.

Correct the code within the program that caused the runaway processing.

⚠ WARNING

If you enter an incorrect PID number, you can kill the wrong program.

If you kill the wrong process, unpredictable controller operation may result which can cause serious injury or death to personnel, and/or damage to equipment.

Be sure that you have the correct PID number before you kill the program.

4.6 Removing a Program

Use the **Remove** button to de-install an RDT program. A program that has been removed no longer begins executing automatically when the system transitions to the Operate state. To remove an RDT program, click on the program name, and select the **Remove** button on the RDT Program Administration dialog box (Figure 4-8). The system prompts you to confirm your choice and then removes your program.

If the program is running, you must halt the program before removing it.

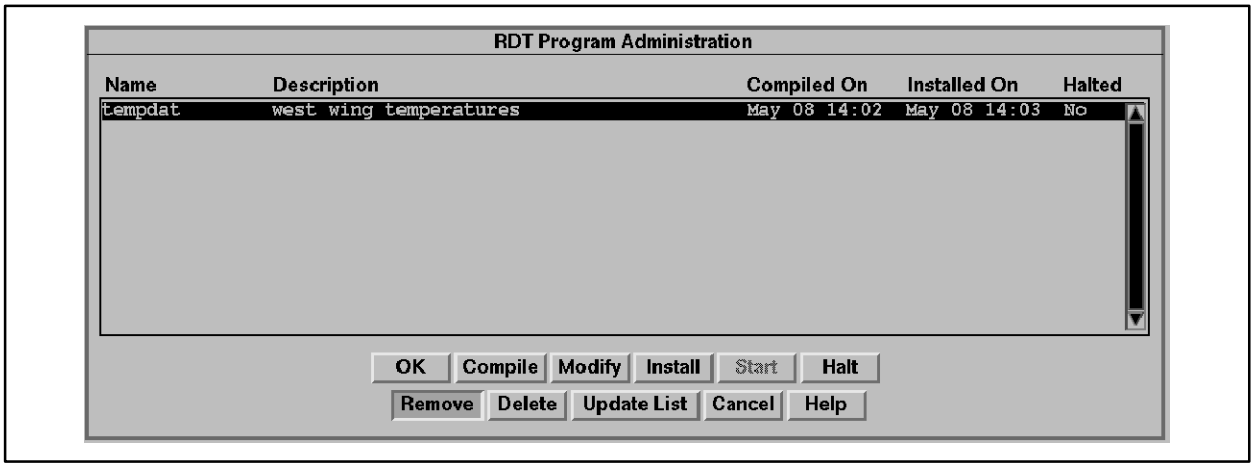


Figure 4-8 Removing an RDT Program

4.7 Deleting a Program

Use the **Delete** button to delete a compiled RDT program. The source code for that program is unaffected by the Delete feature. To delete the file containing the source code, use the operating system **rm** command.

To delete an RDT program, click on the program name, and select the **Delete** button on the RDT Program Administration dialog box (Figure 4-9). The system prompts you to confirm your choice and then deletes your program.

If the program is running, you must halt the program before deleting it.

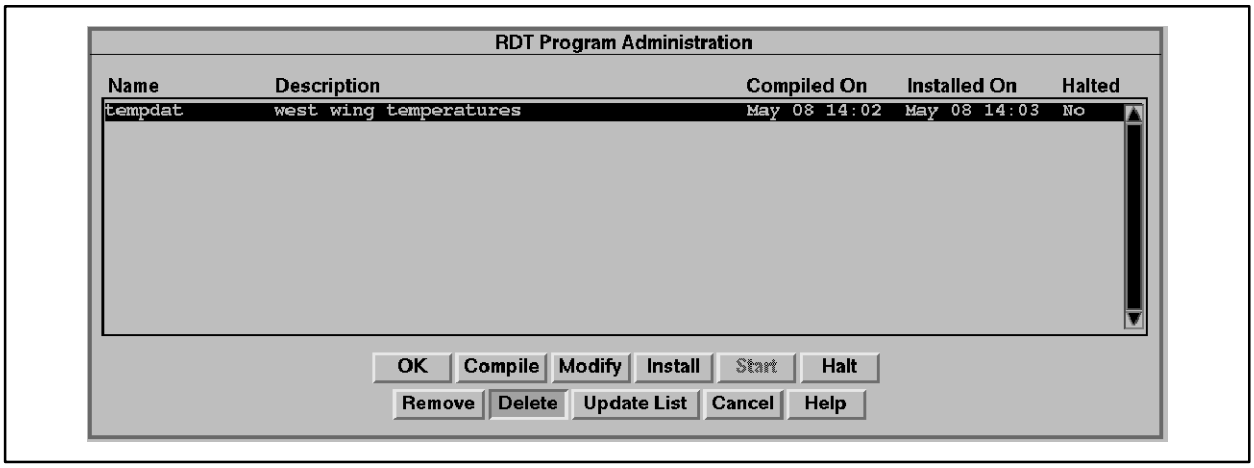


Figure 4-9 Deleting an RDT Program

4.8 Updating the RDT Program List

Use the **Update List** button to update the list of RDT programs in the RDT Program Administration dialog box. This feature causes the program to rebuild its list to reflect changes to RDT programs made on other supervisory stations. You know that the list needs updating if you select a program and find that the buttons are not enabled or disabled correctly for the list entry. For example, if the **Remove** button is enabled but the Installed On column for the selected program is empty, then the list needs updating.

To update the list, select the **Update List** button on the RDT Program Administration dialog box (Figure 4-10).

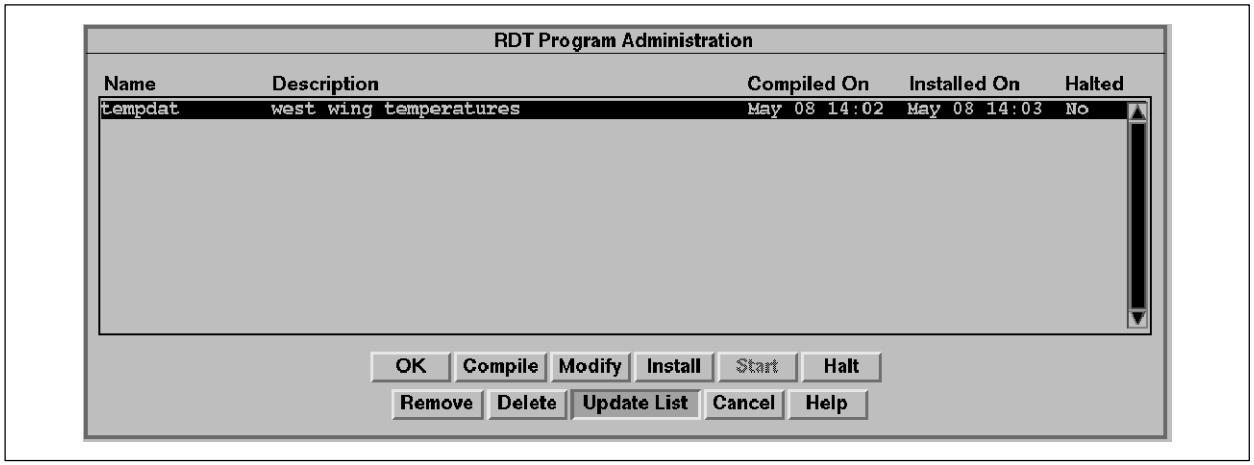


Figure 4-10 Updating the RDT Program List

4.9 Transferring RDT Programs between OSx Stations

If you have dedicated an OSx station to the development of customized files, such as RDT programs, and the station is not part of the multiple-station system, you can install the files for execution on the primary of the multiple-station system. To transfer RDT programs, follow these procedures.

Transfer Program to Media To copy the program on the source station to a diskette, follow these steps.

1. Open an XTerm window and log in as `root`.
2. Change the default directory to the RDT source file directory by entering the following command:

   ```
   cd /usr/tistar/hist/rdt/source
   ```

3. Insert a diskette into the disk drive.
4. Transfer the RDT source file to the diskette. RDT source files have the **.dt** extension. Enter the following command:

   ```
   ls <RDT source file> .dt | cpio -ovc > /dev/rfd0
   ```

   where **<RDT source file>** is the name of the RDT source file. For example, to copy a program called **rdt1** to diskette, enter the following:

   ```
   ls rdt1.dt | cpio -ovc > /dev/rfd0
   ```

5. Remove the diskette and close the XTerm window.

Load Program into Primary Now that your program has been copied to a diskette, you can load it into the primary. Follow these steps.

1. Insert the diskette into the primary.
2. Open an XTerm window and log in as `tistar`.
3. Change the default directory to the RDT source file directory by entering the following command:

   ```
   cd /usr/tistar/hist/rdt/source
   ```

4. To copy the program files from the diskette to the primary, enter the following command:

   ```
   cpio -ivcum < /dev/rfd0
   ```

5. Remove the diskette, and close the XTerm window.

Chapter 5

Debugging Data Transfer Programs

5.1 Troubleshooting Problems during Compile and Installation 5-2
Using the Data Transfer Log File 5-2
Detecting Problems during Compile or Installation 5-3
Testing the Data Transfer Program before Installation 5-4

5.2 Troubleshooting Runtime Problems 5-6
Synchronizing in the Operate State 5-6
Using the Error Log File to Find a Problem 5-6
Sample OSx Log File Entry 5-6
Loss of Communication with Remote Database 5-7
Lost Connection Alarming 5-8
File System Alarm Goes into High High Alarm 5-8

5.3 Tracing the Operation of a Data Transfer Program 5-10
Sample Trace Outputs 5-10

5.1 Troubleshooting Problems during Compile and Installation

CAUTION

In-depth debugging of data transfer programs requires knowledge and experience with the Linux operating system and with C programming.

Failure to observe proper procedures and precautions could result in loss of data.

Only trained personnel should attempt to debug data transfer files.

Using the Data Transfer Log File

OSx stores the various files used by the remote data transfer option in several different directories. Some directories contain working programs; others contain temporary or intermediate files. Figure 5-1 illustrates the directory structure used by the remote data transfer option.

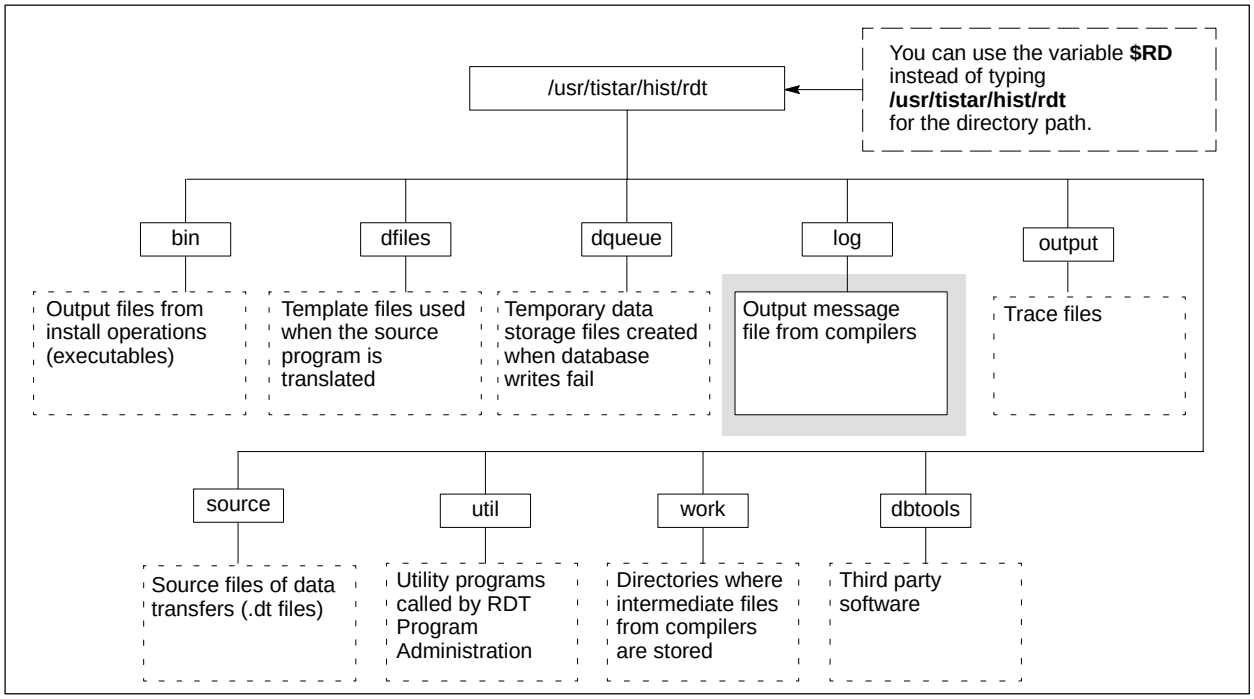


Figure 5-1 Administrative Log File

Detecting Problems during Compile or Installation

OSx creates a file of messages generated while the executable data transfer program is being created. Figure 5-2 shows the stages of compiling a data transfer specification file and the stages that write messages to the administrative log file in the **/usr/tistar/hist/rdt/log** directory.

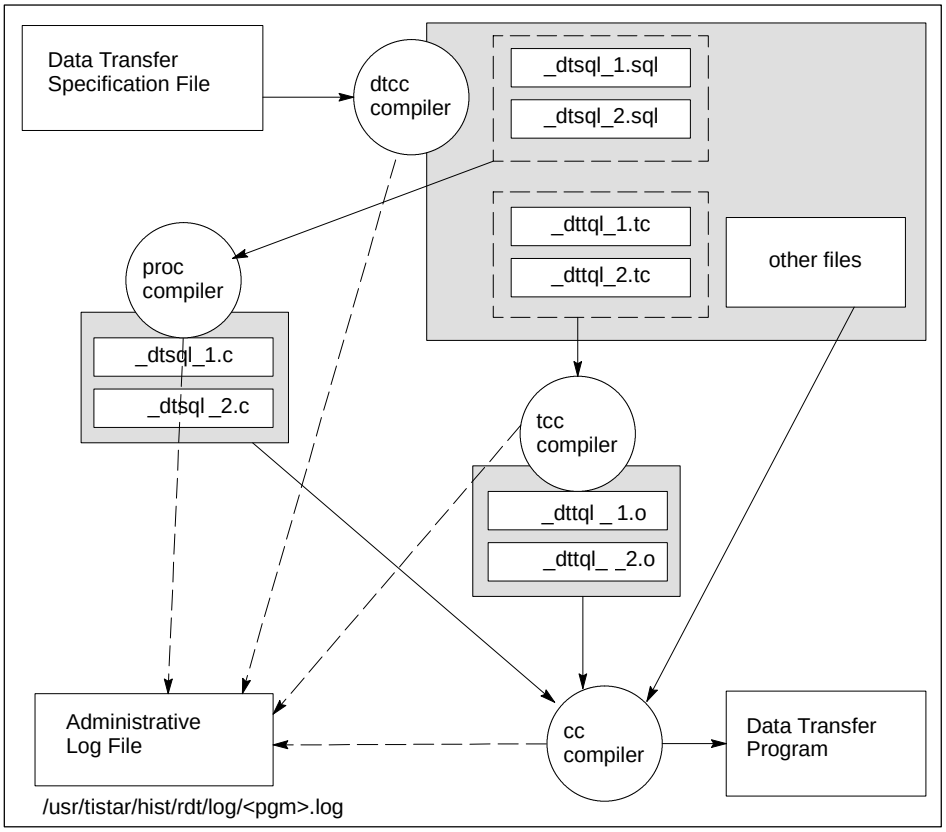


Figure 5-2 Creating the Administrative Log File

Testing the Data Transfer Program before Installation

Before installing a compiled program for normal operation, test the transfers and their events to assure that the program will work as intended.

⚠ WARNING

An online data transfer program can alter the OSx database, thereby affecting values in the controller and/or causing OSx to operate unexpectedly.

Unexpected operation of the controller can cause serious injury or death to personnel, and/or damage to equipment.

To prevent this risk, ensure that OSx is not interacting with the controller before you continue this procedure.

In order to test an RDT program without activating any controlled machinery, use either of the following approaches.

- Dedicate a controller that is not connected to your factory process for testing RDT programs. Use a tool like APT or TISOFT to alter controller values in the controller manually.
- Replace the I/O modules in the controller with I/O simulators.

Use the following procedure to test RDT programs.

1. Install the RDT program with trace output enabled. To enable trace, select the **Modify** button on the RDT Program Administration dialog box. Enter a non-zero value in the Trace Lines field.

2. Set the system to the Operate state. The program begins to execute when the system enters the Operate state.

3. Activate the transfer events and verify that the data transfers occur as planned. You can trigger database events either by changing controller values directly or by changing values on tag details. You cannot force time events.

4. Check the OSx standard error log, located in the **/usr/tistar/data/log.out** file, and the program's trace output file, which is located in the **/usr/tistar/hist/rdt/output/<program_name> .trc** file, to determine if any errors occurred.

5. To stop the program, either set the system to the Offline state, or select the **Halt** button on the RDT Program Administration dialog box.

Correct any execution errors, and then compile and run the program again. If you do not find any errors, the program is ready for final installation.

To minimize interaction between data transfer programs, install and debug one program at a time.

5.2 Troubleshooting Runtime Problems

Synchronizing in the Operate State

You may observe some delay in the execution of an RDT program in the Operate state when another node is synchronizing. To see if this is the cause of the delay, check Network Status, which shows when nodes are synchronizing. After synchronization has finished, program execution will be complete shortly thereafter.

Using the Error Log File to Find a Problem

Runtime errors are generated by either the remote data transfer software or by the remote database. These errors are logged into the Error Log file (**/usr/tistar/data/log.out**). Refer to Table A-12 in Appendix A for the error messages generated by either the remote RDBMS or by OSx, classified by the type of response required.

Sample OSx Log File Entry

Example: The error log file contains the following information:

error 4803 in xread xread ::
ORA-01403: no data found

[_dtsql_5.c:252]

- **ORA** indicates that the message was generated by the Oracle database.
- Message **01403** means that no data was found.
- The error occurred in the data transfer program named **xread**.
- **[_dtsql_5.c:252]** indicates that the error message was generated during a database access within transfer group **5**, specifically the subroutine call on line **252**. Refer to the lines prior to **252** of **_dtsql_5.c** to determine the statement that generated the error.

Loss of Communication with Remote Database

When you are using the RDT feature, and you lose communication with the remote database, the system takes data that it usually writes to the remote database and buffers it on the primary and on the backup. If you change the primary role to a station other than the backup before communication to the remote database is restored, you will lose the buffered data. This also means that you cannot change a station with the sysadmin role to primary during this situation without losing the buffered data.

After the system buffers data on the primary, the RDT program waits until the system copies this data to the backup. Because of the wait, a transfer group may not repeat as frequently as expected.

An example is a transfer group that writes data to a remote database table and has this trigger:

At now repeat every 2 seconds

If you lose connection with the remote database so that the program has to buffer data, and copying the data to the backup takes four seconds, then the transfer group cannot repeat every two seconds. For this example, the transfer group will probably repeat every six seconds.

After you resolve the remote database connection problem, and the system is able to write the buffered data, normal operation should resume.

Lost Connection Alarming

When an RDT program loses connection with the remote database, the program transitions from a “connection good” to a “connection failed” state. When this transition occurs, the following events take place:

- The COMM FAIL bit in the RDT_PGMS:STATUS attribute for the program is turned on.
- The COMM FAIL bit in the SYSTEM:STATUS attribute for the _DATA_XFR system tag is turned on. This triggers a critical alarm.
- An error message is logged to the standard error file, the system message line, and the log printer.

When an RDT program in a “connection failed” state re-establishes connection with the remote database, the program transitions to a “connection good” state. When this transition occurs, the following events take place:

- The COMM FAIL bit in the RDT_PGMS:STATUS attribute for the program is turned off.
- If no other program has its RDT_PGMS:STATUS COMM FAIL bit on, then the COMM FAIL bit in the SYSTEM:STATUS attribute for the _DATA_XFR system tag is turned off.
- A message is logged to the standard error file and the log printer.

File System Alarm Goes into High High Alarm

For PCS software Release 3.1.0 and later: During normal operation, an Oracle log file called **/usr/tistar/hist/rdt/dbtools/oracle/network/sqlnet.log** is created. This file logs all of the Oracle network transactions. The Oracle log file is written to until it reaches the maximum file size of 10 Mbytes. When the file reaches its maximum size, no additional messages can be logged. There is no way to disable the logging of these Oracle messages, and the file system alarm, which monitors available disk space, goes into high high alarm.

Some suggestions for dealing with this situation are listed below.

- You can create a report to truncate the Oracle SQL*Net log file on a regular basis. In cell A1, enter the following (all on one line):

  ```
  @cusmid (mysyscall, "echo NULL >
     /usr/tistar/hist/rdt/dbtools/oracle/network/sqlnet.log")
  ```

  Then schedule the report to run every 24 hours, or at whatever frequency you feel is necessary to keep the **sqlnet.log** file from growing too large.

- You can prune the **sqlnet.log** file selectively using a text editor such as vi.

For earlier releases of PCS software: During normal operation, the **sqlnet.log** file is created by default in the root directory **/usr/tistar/bin**. Since the **hist** slice has more room than the **root** slice, you can alleviate some of the disk space problem by creating a file called **sqlnet.ora** in the following directory:

/usr/tistar/hist/rdt/dbtools/oracle/network/admin

Using a text editor such as vi, edit the **sqlnet.ora** file. Type the following line at the top of the file and save the file:

```
LOG_DIRECTORY_CLIENT = /usr/tistar/hist/rdt/dbtools/oracle/network
```

This file directs the **sqlnet.log** file to be written to the following file:

/usr/tistar/hist/rdt/dbtools/oracle/network

You may still need to truncate the file on a regular basis or prune it selectively. Follow the procedures in the two bullets above. If you leave the **sqlnet.log** file in the **/usr/tistar/bin** directory, you can truncate it there. In cell A1, enter the following:

```
@cusmid (mysyscall, "echo NULL > /usr/tistar/bin/sqlnet.log")
```

Then schedule the report to run every 24 hours, or at whatever frequency you feel is necessary to keep the **sqlnet.log** file from growing too large.

5.3 Tracing the Operation of a Data Transfer Program

When you run a data transfer program the first time, install the program with the trace option (Section 4.3). A data transfer program that runs with the trace option writes trace messages to an output file when specific activities occur. You then read the trace output file to determine whether the program operated as expected.

NOTE: If you enable the trace option, the data transfer program does not execute as quickly as normal because OSx must write trace messages to the output file.

Sample Trace Outputs

Example 1: A transfer group has a START event with an IF clause qualifier and 3 transfers in the group. At 9:00, a START event is received—but the IF clause tests false, so the transfer is not performed. At 10:00, another START event is received, and the IF clause tests true, so the transfers are performed.

```
06/11/92 09:00:22 : START event received for transfer #1
06/11/92 09:00:22 : Result of condition test is FALSE
06/11/92 10:00:40 : START event received for transfer #2
06/11/92 10:00:40 : Result of condition test is TRUE
06/11/92 10:00:40 : Start of xfr_grp #2, xfr #1 READ constant
06/11/92 10:00:40 : End of xfr_grp #2, xfr #1 READ constant
06/11/92 10:00:40 : Start of xfr_grp #2, xfr #2 READ constant
06/11/92 10:00:40 : End of xfr grp#2, xfr #2 READ constant
06/11/92 10:00:40 : Start of xfr_grp #2, xfr #3 READ constant
06/11/92 10:00:40 : End of xfr_grp #2, xfr #3 READ constant
06/11/92 10:00:40 : Start of remote DB LOCK for xfr grp #2
06/11/92 10:00:40 : Start of xfr_grp #2, xfr #2 WRITE (remote)
06/11/92 10:00:41 : End of xfr_grp #2, xfr #2 WRITE (remote)
06/11/92 10:00:41 : Start of xfr_grp #2, xfr #3 WRITE (remote)
06/11/92 10:00:41 : End of xfr_grp #2, xfr #3 WRITE (remote)
06/11/92 10:00:41 : Start of xfr_grp #2, xfr #1 WRITE (local)
06/11/92 10:00:41 : End of xfr_grp #2, xfr #1 WRITE (local)
```

Example 2: The following sample trace output shows an error upon startup where the connection to the remote database failed. (Error numbers associated with the remote database are described in the appropriate Oracle manual.)

```
06/29/92 08:34:18 : Starting: current_status=IDLE
06/29/92 08:34:19 : Error during Connect - 1017
```

Example 3: The following sample trace output is for a program that has two transfers in a transfer group. The transfer group has a START event, a RUN (or REPEAT) event, and an END event scheduled. The trace output shows the normal sequence of actions in a data transfer program.

- At startup, OSx connects to the remote database, and the START event is installed into the program's internal schedule.
- When the START event (a time event) occurs, all the READ data is put into a queue file, the destination database tables are locked, and data is written to the destination database. Note that all data READs are performed before any of the data WRITEs.
- After the first transfer, the START event is removed from the internal schedule table and the RUN and END events are installed. Transfers only occur when triggered by the RUN event (a time interval). If the END event (another time event) occurs, the data does not transfer.
- At 6:00, the system changes from Operate to Offline. Since this transfer program was configured to run only in Operate, the program terminates when the new state becomes 0x10 (Offline state).

```
06/29/92 08:43:58 : Starting: current_status=IDLE
06/29/92 08:44:00 : Connecting to orac at mt1
06/29/92 08:44:03 : Installed TIME event for transfer #1
06/29/92 08:44:03 : current_status=RUNNING
06/29/92 10:00:01 : START event received for transfer #1
06/29/92 10:00:01 : Result of condition test is TRUE
06/29/92 10:00:01 : Start of xfr_grp #1, xfr #1 READ (local)
06/29/92 10:00:01 : End of xfr_grp #1, xfr #1 READ (local)
06/29/92 10:00:01 : Start of xfr_grp #1, xfr #2 READ (local)
06/29/92 10:00:01 : End of xfr_grp #1, xfr #2 READ (local)
06/29/92 10:00:01 : Start of remote DB LOCK for xfr grp #1
06/29/92 10:00:01 : Start of xfr_grp #1, xfr #1 WRITE (remote)
06/29/92 10:00:01 : End of xfr_grp #1, xfr #1 WRITE (remote)
06/29/92 10:00:01 : Start of xfr_grp #1, xfr #2 WRITE (remote)
06/29/92 10:00:01 : End of xfr grp_#1, xfr #2 WRITE (remote)
06/29/92 10:00:01 : Removed TIME event for transfer #1
06/29/92 10:00:01 : Installed INTERVAL event for transfer #1
06/29/92 10:00:01 : Installed TIME event for transfer #1
06/29/92 18:06:43 : SC_EVENT: old state=20 new state=10
06/29/92 18:06:43 : current_status=IDLE
06/29/92 18:06:43 : terminating
```

Appendix A

Data Transfer Error Messages

A.1 Time/Event Scheduling Error Messages A-2

A.2 Database Error Messages A-4

A.3 Data Type Error Messages A-5

A.4 Constant Value Error Messages A-6

A.5 User Variable Error Messages A-7

A.6 File I/O Error Messages A-8

A.7 General Error Messages A-10

A.8 Internal Error Messages A-11

A.9 Syntax Error Messages A-12

A.10 Data Transfer Runtime Error Messages A-14

A.11 Error Classes A-17

A.12 Disabling Error Text on OSx Station A-18

Changing Error Message Destinations A-18

A.1 Time/Event Scheduling Error Messages

Table A-1 Time/Event Scheduling Error Messages

Error number and message	Cause	Recommended solution
4700 – <num> is an invalid calender date.	The number <num> cannot be used as a day number. Day numbers are limited to values between 1 and 31, inclusive.	Use a valid day number: 1 – 31.
4701 – <num> is an invalid month number.	The number <num> cannot be used as a month number. Month numbers are limited to values between 1 and 12, inclusive.	Use a month number: 1 – 12.
4702 – <num> is the maximum value for the given time unit.	For the given time interval in an event, the maximum amount of time that can be entered is <num>.	Use a time interval that does not exceed <num>.
4703 – <num> is an invalid shift number.	The given shift number <num> is not valid. Only the numbers 1 through 6 may be entered for a shift.	Use a valid shift number: 1 – 6.
4704 – Invalid time interval value '<num>'.	The time interval value <num> can not be used. The value is either 0, too large, or negative.	Use a time interval that is not 0, negative, or greater than the maximum limit.
4705 – Invalid time word: <num>.	An invalid word was used for a time specification. The value <num> represents the token value of the word used. Only the words NOW, NOON, and MIDNIGHT are valid time specification words.	Use NOW, NOON, or MIDNIGHT for the time specification word.
4706 – <tword> value '<num>' contains more than 2 digits.	In a time specification, the <tword> time field (hour, minute, or second) contains more than two digits.	Use a two-digit value in the time field.
4707 – <tword> value '<num>' is not valid (0 to 59).	In a time specification, the <tword> time field (minute or second) contains an invalid number.	Use a valid number (0 – 59) for minutes or seconds in the time field.
4708 – Hour value '<num>' is not valid (0 to 23).	In a time specification, the hour field contains an invalid number, <num>.	Use a valid number (0 – 23) in the hour field.
4709 – Invalid start/end token <num> encountered.	While reading a shift event specification, an unknown term was found instead of START or END. The <num> value is the value of the token found.	Use a valid time or a START or END token.

Table continues on next page.

Table A-1 Time/Event Scheduling Error Messages (continued)

Error number and message	Cause	Recommended solution
4710 – Year:<num>. Only years between 1970 and 2037 are legal.	While reading a time event specification, OSx found an invalid year value. The <num> value is the year that was read.	Use a valid year value: 1970 – 2037.
4711 – Invalid date format. Format: <num>.	While reading a default date format line, OSx found an unknown format term instead of DAY, MONTH, or YEAR.	Use DAY, MONTH, and YEAR for the date format.
4712 – Time/Date string <time_string> is invalid.	The <time_string> cannot be converted into a specific time.	Use a time/date string that can be converted to a specific time.
4713 – AM/PM not allowed with 24 hour clock time.	A time specification that uses the time options AM or PM when the time is greater than 12:00:00 is illegal.	Do not use the AM or PM options if you intend to use 24-hour time.
4714 – Illegal time option found.	While reading a time specification, OSx found an unknown term instead of AM or PM.	Use AM or PM for the time specification.
4715 – Invalid time unit found.	For a time interval event, you chose a time unit (seconds, minutes, etc.) that cannot be used for a time interval.	Enter a time unit that is valid for the interval event: SECONDS, MINUTES, HOURS, DAYS, WEEKS.

A.2 Database Error Messages

Table A-2 Database Error Messages

Error number and message	Cause	Recommended solution
4723 – Alias name <name> has not been defined.	A remote database identifier <name> is used but never defined.	Add the missing alias definition or use a defined alias.
4724 – Attribute <name> is invalid.	The attribute <name> was not found in the OSx database for the relation given.	Verify that all OSx database items are correctly named. The name may need to be enclosed in quotes ('').
4725 – Can't find bitname <name> in OPR_NODES database.	The bit <name> was not found in the OSx database for the given attribute.	Verify that all OSx database items are correctly named. The name may need to be enclosed in quotes ('').
4726 – Name '<name>' is not a legal tag or relation name.	The <name> was not found in the OSx database as either a tag name or as a relation name.	Verify that all OSx database items are correctly named. The name may need to be enclosed in quotes ('').
4727 – Retrieve of tag <name>'s relation ID failed. Error: <num>.	An attempt to read the relation ID for tag <name> failed. The database returned the <num> error code.	If problem cannot be determined by the database error code, see your system administrator for assistance.
4728 – Database error <num> when retrieving <name>.	A database access for <name> failed. The error code returned <num>.	If problem cannot be determined by the database error code, see your system administrator for assistance.
4729 – Error <num> during database access for attribute <name>.	An attempt to read the attribute <name> from the OSx database failed. The error code returned <num>.	If problem cannot be determined by the database error code, see your system administrator for assistance.

A.3 Data Type Error Messages

Table A-3 Data Type Error Messages

Error number and message	Cause	Recommended solution
4735 – No data type defined for <source> or <destination>.	Either a transfer's <source> or its <destination> contain undefined data types.	Add an explicit data type definition to the source and/or destination.
4736 – Remote <table_name>:<column_name> data type is unknown.	A remote database item is an unknown data type. A data type cannot be inferred from the item's usage, and no explicit data type definition exists.	Add an explicit data type definition to the remote database item.
4737 – Illegal transfer to/from a string (<source> and <destination>).	Either the source attribute <source> or the destination attribute <destination> is the data type STRING and the other one is not. A STRING data type cannot be converted to any other data type.	Verify the intent of the transfer and look for an explicit type definition that may be incorrect.
4738 – INFO: remote <table_name>:<column_name> is assumed to be data type <type_name>.	The named remote database item given by <table_name> and <column_name> is assigned the <type_name> data type.	If this type does not agree with how the item has been defined in the remote database, then an explicit data type assignment should be performed.
4739 – WARNING: Type define for <relation>:<attribute> will be ignored.	An explicit data type is defined for the OSx database item named by <relation> and <attribute>. OSx database items already have a data type defined by the database and an attempt to redefine the type is ignored.	No action is required. To eliminate the warning, remove the explicit type definition.
4740 – Data mismatch in '<string>' WHERE clause.	An illegal data type mismatch exists between items in a WHERE clause specification. The <string> is part of the WHERE clause that generates the problem. In some cases, only the name of an attribute involved is given. Typically, the only data type mismatches that can occur are between a STRING data type and any other type.	Verify the data types of the items involved.

A.4 Constant Value Error Messages

Table A-4 Constant Value Error Messages

Error number and message	Cause	Recommended solution
4748 – IF clause: Can't cast string type item.	An illegal data type mismatch exists between items in an IF clause specification. One of the items is defined as a STRING data type and the other one is not.	A string item cannot be compared with a non-string item; verify the data types and usage.
4749 – Illegal relational operator '<num>' type found.	An illegal relational operator token identifier <num> is found. Only the following operators are allowed: **<= >= < > <> ==**	Use a valid relational operator.
4750 – Constant <num> is out of range.	The given constant value <num> is out of range for the assigned data type.	Change the type of the constant or use a valid number.
4751 – Hex value <num> is too large.	The given hexadecimal value <num> is too large (contains too many digits).	Use a valid number.
4752 – Identifier exceeds <num> characters.	A word contains too many characters. <num> is the maximum number of characters that are allowed.	Use a shorter word.
4753 – Unterminated quoted string found: <string>.	The <string> begins with a starting quotation mark but does not end with a terminating quotation mark.	Add the missing quotation mark.
4754 – Illegal bit test value: <constant>.	The bit test value <constant> is illegal. A bit can only be tested against the values **0** (for OFF) or **1** (for ON).	Use only 0 or 1 for a bit test value.
4755 – WARNING: signs in front of hex numbers are ignored.	A hexadecimal value with a leading sign (**+** or **-**) found. The sign was ignored.	Remove the sign from the hexadecimal value.
4756 – Quoted string is too large for buffer.	While trying to read a quoted string, the program ran out of buffer space. String length must be less than 256 characters.	Use a shorter string.
4757 – Illegal data type for constant <constant>.	The constant <constant> is an illegal data type.	Review statement syntax for errors.

A.5 User Variable Error Messages

Table A-5 User Variable Error Messages

Error number and message	Cause	Recommended solution
4761 – User variable <num> used but not set.	The user variable <num> is read but its data is not written to it yet.	Add a data write statement to the user variable before trying to read the user variable.
4762 – User variable <num> loaded more than once.	Data written into user variable <num> more than once within the same transfer group.	Only write data to a user variable once within each transfer group.
4763 – User variable ID must be in the range 0 and <num>.	The value for a user variable identifier must be a number in the the range of 0 to <num>.	Use a valid user variable identifier.
4764 – User variable <num> data type is unknown.	The user variable <num> data type is unknown and a type cannot be assumed from its usage.	Add an explicit data type definition to the user variable.
4765 – User variable <num> type is defined differently.	The user variable <num> is defined as having two different data types within the same transfer group.	Verify the usage of the user variable and the data type definition.

A.6 File I/O Error Messages

Table A-6 File I/O Error Messages

Error number and message	Cause	Recommended solution
4769 – Can't open <name> file. Error: <num>.	The file <name> could not be opened. The Linux error code is <num> (see Table A-7).	See your system administrator for further assistance.
4770 – Can't close <name> file. Error: <num>.	The file <name> could not be closed. The Linux error code is <num> (see Table A-7).	See your system administrator for further assistance.
4771 – Can't update <name> file. Error: <num>.	The file <name> could not be updated. The Linux error code is <num> (see Table A-7).	See your system administrator for further assistance.
4772 – Error in creating <name> file. Error: <num>.	The file <name> could not be created. The Linux error code is <num> (see Table A-7).	See your system administrator for further assistance.
4773 – Error in initializing <name> file. Error: <num>.	The file <name> could not be initialized. The Linux error code is <num> (see Table A-7).	See your system administrator for further assistance.

Table A-7 Linux Error Codes and Possible Causes

Error code	Cause	Error code	Cause
2	File does not exist or a directory in the pathname to the file does not exist.	21	Named file is a directory.
4	Signal from operating system interrupted the file I/O operation.	23	System file table is full.
13	Permission denied.	24	Process has too many open files.
14	Bad address.	26	File is busy.
17	File already exists.	30	File system is read-only.
20	Part of the pathname to the file is not a directory.		

A.7 General Error Messages

Table A-8 General Error Messages

Error number and message	Cause	Recommended solution
4779 – Error in statement ending in line #<num>, possibly near '<string>'.	This message appears for most programmer errors. Since several lines of the source program may be read before error checking is performed, the error may occur before the actual line number given by <num>, and it may be located somewhere before the word <string>. Where possible, one or more messages may be given to further pinpoint the error.	If no other error message occurs, then the error is a syntax error. This means that OSx found a word or symbol where it did not belong. Possible causes are missing semicolons after statements, incorrect parentheses, misspelled key words, and missing key words. Carefully review the line identified by <num> and previous lines to ensure that the correct statement syntax is used.
4780 – ERROR: Invalid number of arguments.	The Remote Data Transfer pre-compiler program **dtcc** has been invoked incorrectly. This error will not happen if the administration tool **dt_admin** is used to compile a data transfer source file.	Contact the system administrator.
4781 – ERROR: Source file name must be 7 characters or less.	All data transfer source programs names must be limited to seven characters, not counting the **.dt** extension.	Change the name of the data transfer source program.
4782 – Given remote DB type not installed.	Wrong DB type chosen for remote DB.	Correct the DB type (ORACLE_NET8 or ORACLE_SV2 for Oracle).

A.8 Internal Error Messages

The errors listed in Table A-9 are caused by failures within the pre-compiler program itself or in the operating system. If the problem is caused by something wrong with the operating system, return the operating system to normal before re-compiling. If the error is internal to the pre-compiler, try rewording the source file being compiled so that the pre-compiler can avoid the condition that is causing the problem. In most cases the error should be reported to your Siemens representative. In the U.S.A., call 800–333–7421; outside the U.S.A, call 49–911–895–7000.

Table A-9 Internal Error Messages

Error number and message
4789 – INTERNAL ERROR: <fname>: <subroutine> failed. Error: <num>.
4790 – INTERNAL ERROR: <fname>: Unknown event phrase ntype: <num>.
4791 – INTERNAL ERROR: <fname>: Null node pointer.
4792 – INTERNAL ERROR: <fname>: Invalid node: <num>.
4793 – INTERNAL ERROR: <fname>: WHERE node missing.
4794 – INTERNAL ERROR: <fname>: <pname> pointer going to be overwritten.
4795 – INTERNAL ERROR: <fname>: Illegal data type: <num>.
4796 – INTERNAL ERROR: <fname>: Null data type: <num>.
4797 – INTERNAL ERROR: Invalid DB update option.
4798 – INTERNAL ERROR: <fname>: <relation> attribute node missing.
4799 – INTERNAL ERROR: <fname>: Relation node missing.
4800 – INTERNAL ERROR: <fname>: Null string pointer found.

A.9 Syntax Error Messages

Table A-10 Syntax Error Messages

Error number and message	Cause	Recommended solution
4807 – <relation>:<attribute> is not qualified with a WHERE clause.	A database item required a WHERE clause that specified a row in the database.	Add a WHERE clause so that only one row in the referenced <relation> is identified.
4808 – Illegal left operand in WHERE clause.	In a WHERE clause with the form "WHERE a:c = 5", something is wrong with the "a:c" part of the clause.	Verify the wording of the WHERE clause.
4809 – Missing WHERE clause text for RETRIEVE.	A data transfer that is read from the OSx database does not have a WHERE clause.	Add a WHERE clause so that only one row in the referenced relation is identified.
4810 – WHERE clause not allowed with <name>.	A WHERE clause is used where one is not allowed. This typically happens when an OSx tag name specifies a database item.	Remove the WHERE clause causing the error.
4811 – Incorrect syntax somewhere after DEFAULT.	An unexpected word or symbol was found in DEFAULT statement.	Check the wording of the DEFAULT statement.
4812 – Mismatch in the number of READ/WRITE data points.	A multiple attribute transfer was found where the number of attributes to be read does not match the number of attributes to be written. "tbl1:a,b,c,d -> r1!tbl2:a,b"	Change all multiple attribute transfers so that they use the same number of read and write attributes.
4813 – Only "=" test is allowed on bit names.	An IF clause with a bit test condition was found where the bit test operator is not **=**.	Change all bit test operators to **=**.
4814 – A maximum of 100 attributes allowed per transfer.	A multiple attribute transfer was found that tries to read/write more attributes than is allowed. The maximum number of attributes to be read (or written) in a single transfer is 100.	Verify that each data transfer reads no more than <num> attributes.
4815 – Unterminated comment.	An unterminated comment was found in the source program.	Add the ***/** to the end of the comment.

Table continues on next page.

Table A-10 Syntax Error Messages (continued)

Error number and message	Cause	Recommended solution
4816 – OPR_NODES tag DB item must be used in a status write phrase.	OSx found a “status write” for a transfer group, and the database item receiving the status is not a OSx tag name.	Fix the status write to use an OSx tag.
4817 – Remote <word> is blank.	The required <word> field in a remote database alias definition statement is either blank or illegal.	Verify the correctness of the alias definition statement.
4818 – Remote DB item host name is missing.	A remote database item is referenced, but the remote database alias name is missing. All remote database items must be preceded by the alias “<alias_name>!”.	Fix the remote database item reference.
4819 – Transfer destination is invalid or missing.	A data transfer did not specify a data write location.	Add the missing data write phrase to the data transfer.
4820 – Transfer source is invalid or missing.	A data transfer did not specify a data read location.	Add the missing data read phrase to the data transfer.
4821 – Too many transfer groups. Only 100 groups allowed.	The source file contains too many transfer groups. There is a maximum of 100 transfer groups allowed per program.	Remove some of the transfer groups.
4822 – File transfers not supported at this time.	A file transfer request found.	Remove the file transfer from the program source file.
4823 – Attribute name for ’<name>’ is missing.	OSx found a database item reference for relation <name>, but no attribute was named. All database item references must have a relation (or table) and an attribute (or column) name.	Add the missing attribute name.
4824 – Only 50 transfers per transfer group allowed.	Transfer group contains too many transfer statements. There is a limit of 50 transfers per transfer group.	Remove some of the transfers from the group.
4825 – Only one remote DB definition allowed.	More than one remote DB alias definition found.	Remove the extra definitions.

A.10 Data Transfer Runtime Error Messages

These messages listed in Table A-11 are produced by a running data transfer process and logged to the OSx error log.

Table A-11 Data Transfer Runtime Error Messages

Error number and message	Cause	Recommended solution
4884 – Can't find process's executable file.	This error occurs only when a process is manually started. The process searched the standard data transfer bin directory and the appropriate "work" directory and cannot find its executable file.	Verify that the executable transfer program file exists in the **$RD/bin** and **$RD/work/<pg_nam>** directories.
4885 – <process_name>: transfer group #<num> remote DB lock failed.	The data transfer program <process_name> tried to lock a table in the remote database for writing and the lock failed. The lock is for the data write associated with the transfer at line number <num> in the program source file.	Verify that no other user has locked the destination table in the remote database.
4886 – <process_name>: transfer group #<num> local DB lock failed.	The data transfer program <process_name> tried to lock a table in the OSx database for writing and the lock failed. The lock is for the data write associated with the transfer at line number <num> in the program source file.	This error may occur occasionally as data write locks on OSx tables collide. If this error happens continually, then steps must be taken to slow down the operation or to increase the timing between the data writes by different processes.
4887 – <process_name>: Queue file limit reached for group #<num>.	The data transfer program <process_name> saved transfer data in a queue file and ran out of file space. No more data can be saved.	Resolve the error that is causing the data transfer for transfer group number <num> so that the saved data can be transmitted.
4888 – <process_name>: NULL pointer found (internal err).	The data transfer program <process_name> encountered a null pointer in its internal processing.	Shut down and restart the data transfer process. Contact your system adminstrator.
4889 – Value truncated. Original length = <num>.	A value transferred to or from the remote database is too long. The Oracle code truncated the value to fit into the local machine's storage register. This error occurs when the data type of the remote database attribute is incompatible with the data type of the OSx database item.	Verify that the data types for all remote database items are equivalent to the corresponding OSx database items.

Table continues on next page.

Table A-11 Data Transfer Runtime Error Messages (continued)

Error number and message	Cause	Recommended solution
4890 – <process_name>: Invalid database event received.	The data transfer program <process_name> received an unexpected OSx database event. This should not happen during normal operation.	Verify that all data transfers are occurring as planned. If not, shut down and restart the data transfer program. If this error occurs repeatedly, contact your system administrator.
4891 – <process_name>: Re-scheduling missed time event.	A scheduled time event did not occur. This error occurs at process startup when a past time event is found. In this case, normal operation proceeds. If the error occurs at any other time, then either the system is overextended (too many operations ongoing) or a time schedule is too frequent.	Schedule your transfers to occur less frequently, or improve the efficiency of transfers by providing an indexed key attribute in your table.
4892 – <process_name>: Event table out of space (internal err).	The data transfer program <process_name> ran out of space in its internal event table.	If this error occurs, shut down and restart the data transfer process. If this error occurs repeatedly, contact your system administrator.
4893 – <process_name>: Switch invalid case (internal err).	The data transfer program <process_name> is operating incorrectly.	Shut down and restart the data transfer process.
4894 – <process_name>: No active states for data xfr process.	The data transfer program <process_name> is not active in any system state. No data transfers occur.	Stop the process, reinstall it using the dt_admin tool, and then change system state to the configured active state.
4895 – <process_name>: Invalid number of arguments.	The data transfer program <process_name> invoked with missing or extra arguments. This error occurs only when you manually start the process.	Start the data transfer program properly.
4896 – <RDBMS_error_message>.	This OSx error reports errors from the remote database subsystem. The error number of the remote database error appears in the message.	Refer to the appropriate Oracle manual for further information.
4897	This error triggers an OSx information-type alarm for the remote data transfer system tag _DATA_XFR.	No action.

Table continues on next page.

Table A-11 Data Transfer Runtime Error Messages (continued)

Error number and message	Cause	Recommended solution
4898	This error triggers an OSx degraded-type alarm for the remote data transfer system tag _DATA_XFR.	No action.
4899	This error triggers an OSx critical type alarm for the remote data transfer system tag _DATA_XFR.	No action.

A.11 Error Classes

Table A-12 Error Classes

Error Class	Description	Error Code	
1	No error message generated. No alarm generated. No data queued.	Oracle	0 no error 1075 already logged on
2	Generates an alarm. Ignored if a **read status** was configured for the transfer group experiencing the error.	OSx	249 data not found Any error code not listed elsewhere in this table.
		Oracle	1403 data not found Any error code not listed elsewhere in this table.
3	Generates an alarm. Queues data for transfers writing data to remote database.	Oracle	17 maximum calls exceeded 18 maximum sessions exceeded 19 maximum sessions exceeded 20 maximum processes exceeded 52 maximum queue exceeded 53 maximum queue exceeded 54 resource busy and NOWAIT 59 maximum files exceeded 60 deadlock 700 maximum cache exceeded 702 maximum cache exceeded 1154 database busy 1155 database busy 1156 recovery in progress 1536 space quota exceeded 1542 tablespace offline 1574 maximum transactions exceeded 1575 timeout on space resource 1631 max extents reached 1653 unable to extend table 1654 unable to extend index 1655 unable to extend cluster 1656 max extents reached 1659 unable to allocate 1683 unable to extend index 1684–1685 max extents reached 1686 max files reached 1950 no privileges

This table is continued on the next page.

Table A-12 Error Classes (continued)

Error Class	Description	Error Code		
4	Generates an alarm. Queues data for transfers writing data to remote database. Stops access until reconnected to remote database.	Oracle	470	LGWR process terminated
			471	DBWR process terminated
			472	PMON process terminated
			473	ARCH process terminated
			474	SMON process terminated
			1012	not logged on
			1014	shutdown in progress
			1033	database change in progress
			1034	database not available
			1071	database not started
			1089	shutdown in progress
			1090	shutdown in progress
			1092	database terminated
			1573	database shutting down
			3113	EOF on communication channel
			3114	not connected
			6000–6499	SQL*NET errors
			6600–6999	SQL*NET errors
			12150	unable to send data
			12151	bad packet type
			12152	unable to send break
			12153	not connected
			12203	unable to connect
			12535	operation timed out
			12541	no listener
			12543	destination host unreachable
			12545	connect failed
			12546	permission denied
			12547	lost contact
			12548	incomplete read or write
			12549	resource quota exceeded
			12552	operation interrupted
			12554	operation in progress
			12555	permission denied
			12560	protocol adapter error
			12561	unknown error
			12564–12583	TNS errors
			12612	connection busy

A.12 Disabling Error Text on OSx Station

Changing Error Message Destinations

At every occurrence of a remote database error, the error text returned by the code is written into the standard OSx error file **/usr/tistar/data/log.out** and is displayed on the system message line of all OSx stations. This is the factory-set error action and is set on all shipped SIMATIC PCS 7 OSx systems. If you do not want the error messages to go to the OSx station displays, change the destination bit mask in the ERR_MSGS relation for **errno 4896**. To change it, follow the steps below.

1. Create a report that contains a cell with the following content:

   #write (err_msgs : err_action ; err_msgs : errno=4896 : 0x800)

2. Compile the report.

3. Schedule the report to run once.

Once this item has been changed in the OSx database, you only have to repeat it if the database is rebuilt.

NOTE: Be aware that in some cases a _DATA_XFR system alarm is generated only for the first occurrence of a remote database error. Disabling the OSx station message line display for **errno 4896** may tend to mask recurring errors.

Appendix B

Using Attributes to Trigger Events

In the standard OSx database, some attributes cannot be used as an event trigger. Table B-1 lists the no-event attributes for the process I/O tables.

Table B-1 OSx No-Event Attributes

Table	No-Event Attributes
AI	pseudo_tag, units, change
AO	pseudo_tag, units, change
CALC	pseudo_tag, units, change
CTR	pseudo_tag, units, l_range
DEVICE	pseudo_tag
DI	pseudo_tag
DO	pseudo_tag
DI10	pseudo_tag
DO10	pseudo_tag
IVAR	pseudo_tag, units, l_range, h_range
LOOP	pseudo_tag, units, change
TEXT	pseudo_tag, units
TMR	pseudo_tag, units, change

Appendix C

Quick Reference for Instruction Syntax

A widely used notation for specifying the syntax of a programming language is the Bachus-Naur Form (BNF). The BNF is a context-free grammar notation which consists of the following parts:

- A set of tokens, known as terminal symbols.
- A set of nonterminals.
- A set of productions, where each production consists of a non-terminal, called the left side of the production, and a sequence of tokens (terminal symbols) and/or nonterminals, called the right side of the production. A colon separates the two sides.

An sample production is shown below:

run_phrase : REPEAT ON EVENT simple_db_item CHANGES

By substituting the phrase “can have the form of” for the colon and quoting the nonterminals, the statement becomes:

“run_phrase” can have the form of

REPEAT ON EVENT “simple_db_item” CHANGES

The capitalized words are the tokens, or key words, and must be entered as shown. A production may have more than one form. The pipe symbol (|) is used to indicate an alternative form. For example, the production:

```
time_unit : SECONDS
          | MINUTES
          | HOURS
```

means

“time_unit” can have the form SECONDS or MINUTES or HOURS

Quick Reference for Instruction Syntax (continued)

The following BNF shows the syntax of the instructions used for the OSx Remote Database Interface.

```
[ ]       Brackets  indicate an optional item
|         Bar       indicates selection (either, or)
{ }       Braces    indicate one or more items
'symbol'            indicates a control symbol which must be entered exactly as it
                    appears
< >       Angles    indicate user-entered data
/* text */          indicates a comment
UPPERCASE           indicates a keyword
```

NOTE: String constants for ORACLE database items must be enclosed within single quotation marks. Example: `WHERE(w!tab:cd='NAME')`. OSx database item names that contain spaces must be enclosed in double quotation marks. Example: `IF(LOOP_0:STATUS:"H ALARM"=1)`

```
program : { sentence }

sentence
   : default_define ';'
   | remote_define ';'
   | transfer_define ';'

default_define
   : DEFAULT update_type
   | DEFAULT DAY MONTH YEAR
   | DEFAULT MONTH DAY YEAR
   | DEFAULT YEAR MONTH DAY
   | DEFAULT YEAR DAY MONTH

update_type
   : APPEND
   | INSERT
   | REPLACE
   | UPDATE

remote_define
   : <alias> IS db_type DBMS <userid> ':' <password> ':' <machine> ':' <db>
   | <alias> IS <userid> ':' <password> ':' <machine> ':' <account>
   | <alias> IS <userid> ':' <password> ':' <machine>
```

```
db_type
   : ORACLE_NET8
   | ORACLE_SV2

transfer_define
   : event_statement '(' { transfer_statement } ')' [ status_read ]

event_statement
   : start_phrase [ run_phrase ] [ end_phrase ]

start_phrase
   : ON EVENT simple_db_item CHANGES [ IF conditions ]
   | AT time [ date ] [ IF conditions ]

run_phrase
   : REPEAT ON EVENT simple_db_item CHANGES [ IF conditions ]
   | REPEAT EVERY <integer> time_unit [ IF conditions ]

end_phrase
   : UNTIL EVENT simple_db_item CHANGES [ IF conditions ] [ RECYCLE ]
   | UNTIL time [ date ] [ IF conditions ] [ RECYCLE ]

conditions
   : condition_phrase
   | conditions AND conditions
   | conditions OR conditions
   | '(' conditions ')'

condition_phrase
   : if_db_item rel_operator if_db_item
   | if_db_item rel_operator <text>
   | if_db_item rel_operator number_constant
   | if_bit_db_item rel_operator number_constant

if_db_item
   : <rel_name> ':' attribute [ WHERE where_clause ]

if_bit_db_item
   : <rel_name> ':' attribute ':' <bit_name> [ WHERE where_clause ]
   | <rel_name> ':' attribute ':' <hex> [WHERE where_clause]

rel_operator
   : '<='
   | '>='
   | '<>'
   | '='
   | '<'
   | '>'
   ;
```

Quick Reference for Instruction Syntax (continued)

```
time
   : NOW
   | NOON
   | MIDNIGHT
   | SHIFT <integer> [ START | END ]
   | <integer> ':' <integer> [ ':' <integer> ] [ AM | PM ]
   | <integer> '.' <integer> [ '.' <integer> ] [ AM | PM ]

time_unit
   : SECONDS
   | MINUTES
   | HOURS
   | DAYS
   | WEEKS

date
   : TODAY
   | <integer> '/' <integer> '/' <integer>
   | <integer> '-' <integer> '-' <integer>
   | <integer> ':' <integer> ':' <integer>
   | <integer> '.' <integer> '.' <integer>

transfer_statement
   :destination '<-' source [ update_type ]

source
   : src_db_item
   | <text>
   | number_constant
   | DT_TIME_N
   | DT_TIME_S
   | user_variable

destination
   : dest_db_item
   | user_variable

status_read
   : <tagname> ':' attribute '=' number_constant ':' number_constant
   | <tagname> ':' attribute '=' <text> ':' <text>

simple_db_item
   : <rel_name> ':' attribute  [ WHERE where_clause ]

src_db_item
   : <rel_name> ':' source_list [ WHERE where_clause ]
   | <alias> '!' <rel_name> ':' source_list [ WHERE remote_where_clause ]
```

```
source_list
   : src_list_item
   | src_list_item { ’,’ src_list_item }

src_list_item
   :attribute
   |DT_TIME_N
   |DT_TIME_S
   |<text>
   |number_constant
   |user_variable

dest_db_item
   :<rel_name> ’:’ attribute_list [WHERE where_clause]
   |<alias> ’!’ <rel_name> ’:’ attribute_list [WHERE where_clause]

attribute_list
   : attribute
   | attribute {’,’ attribute}

attribute
   : <att_name>
   | ’[’ type_define ’]’ <att_name>

where_clause
   : where_item
   | where_clause AND where_item

where_item
   : ’(’ <rel_name> ’:’ attribute ’=’ simple_db_item ’)’
   | ’(’ <rel_name> ’:’ attribute ’=’ <text> ’)’
   | ’(’ <rel_name> ’:’ attribute ’=’ number_constant ’)’
   | ’(’ <rel_name> ’:’ attribute ’=’ user_variable ’)’

remote_where_clause
   : remote_where_item
   | remote_where_clause AND remote_where_item

remote_where_item
   : ’(’ <alias> ’!’ <rel_name> ’:’ attribute ’=’ remote_simple_db_item ’)’
   | ’(’ <alias> ’!’ <rel_name> ’:’ attribute ’=’ <text> ’)’
   | ’(’ <alias> ’!’ <rel_name> ’:’ attribute ’=’ number_constant ’)’
   | ’(’ <alias> ’!’ <rel_name> ’:’ attribute ’=’ user_variable ’)’

remote_simple_db_item
   : <alias> ’!’ <rel_name> ’:’ attribute WHERE remote_where_clause
```

```
user_variable
   : '%' <integer>
   : '[' type_define ']' '%' <integer>

number_constant
   :  [ '+' | '-' ] number
   : '[' type_define ']' [ '+' | '-' ] number

number
   : <integer> '.' <exponent>
   | <integer> '.' <integer>
   | <hex>
   | <exponent>
   | <integer>

type_define
   : UINT8
   | UINT16
   | UINT32
   | SINT8
   | SINT16
   | SINT32
   | FLOAT32
   | FLOAT64
   | STRING
```

Appendix D

Decimal/Hex Number Conversion

You can use the bc conversion utility that comes with your system to convert numbers from one format to another. Follow the steps below.

1. Open an Xterm window and log in as `tistar`.

2. To access the calculator process, type `bc` and press **Enter**.
   (There will not be a prompt after this step.)

3. To convert decimal to hexadecimal, type `obase=16` and press **Enter**.
   This sets the output number base to hexadecimal.

   To convert hexadecimal to decimal, type `ibase=16` and press **Enter**.
   This sets the input number base to hexadecimal. Always enter the hex digits A through F in upper case.

4. Type in the number that you want converted.

   Example: If you type `obase=16`, and then type `14`, the system returns the hex number **E**. If instead you type `ibase=16` and then enter `E`, the system returns the decimal number **14**.

5. To end the program, type `quit` and press **Enter**.

Appendix E

Date Formats for Year 2000 and Beyond

To implement the **ora2** and **ora4** Oracle date options correctly, you must configure the date format properly in the Oracle database on the Oracle server host. Recommended settings are described below.

ora2 Oracle Date Option

To use the two-digit year Oracle date mode correctly, ask the database administrator for the Oracle database to make the following setting in the database initialization file:

```
nls_date_format="DD-MON-RR"
```

The RR year designator in this date format entry tells Oracle to use the default century for two-digit year values as follows:

70 – 99 => 1970 – 1999
00 – 47 => 2000 – 2047

If you do not use the RR designator, Oracle interprets the two-digit date of 00 from OSx as 1900.

NOTE: The initialization file entry NLS_DATE_FORMAT sets the default date format for the entire Oracle database. All applications, including OSx, must conform to the specified format.

---

ora4 Oracle Date Option

To use the four-digit year Oracle date mode correctly for the transition from 1999 to 2000, follow the procedure above for **ora2**, but enter the date format entry below:

```
nls_date_format="DD-MON-YYYY"
```

If this entry already exists in the file, no change is needed.

std Oracle Date Option

If you use the default timestamp format, DD/MM/YY HH:MM:SS, the data type definition of the receiving data field must be a character string. You cannot use the Oracle DATE data type format with the **std** option. Therefore, no entry is needed in the Oracle database initialization file.

Index

A

Accessing, RTD Program Administration dialog box, 4-2

Alarm, communications failure, 5-8

Alias instructions
  defined, 2-3, 3-2
  syntax and usage, 3-10

Assigning, data types, 2-7

Attributes, no-event, B-1

B

BNF notation, C-1

C

Changing
  error message destination, A-18
  OSx database tables, 3-18
  RDT program, 4-9

Compile
  debugging, 4-7, 5-3
  operation theory, 4-4
  RDT program, 4-6
  transfer log file, 5-2

Connection mode, RDT program, 4-9

Connection retries, RDT program, 4-10

Constant value error messages, A-6

Conversion, decimal/hex, D-1

Creating
  data transfer file, 2-3
  data transfer program, 2-6
  data transfer specification file
    on non-OSx station, 2-13
    on OSx station, 2-12

Customized file, RDT, transferring, 4-17

D

Data transfer instructions
  defined, 2-3, 3-2
  syntax and usage, 3-12

Data transfer specification file
  alias instructions
    defined, 2-3, 3-2
    syntax and usage, 3-10
  creating
    on non-OSx station, 2-13
    on OSx station, 2-12
  data transfer instructions
    defined, 2-3, 3-2
    syntax and usage, 3-12
  database write actions, 3-8
  date formats, 3-9
  default instructions
    defined, 2-3, 3-2
    syntax and usage, 3-8

Data types
  assigning, 2-7
  error messages, A-5
  explicit assignment, 2-8
  implied assignment, 2-8
  list, 2-7

Database
  error messages, A-4
  OSx, 1-2
  remote, 1-2

Date format, 3-9

De-installing, RDT program, 4-14

Debug
  compile errors, 4-7, 5-3
  runtime errors, 5-6

Decimal conversion, D-1

Default instructions
  defined, 2-3, 3-2
  syntax and usage, 3-8

Defining
an alias, 3-10
defaults for data transfers, 3-8
transfer items, 3-5
Deleting
RDT program, 4-15
table, 3-5
Directory structure, 1-6, 5-2

E

End event, 3-15
Error messages
classes, A-17
constant value, A-6
data transfer runtime, A-14
data type, A-5
database, A-4
debugging compile, 4-7, 5-3
error text, A-18
file I/O, A-8
general, A-10
internal, A-11
runtime, 5-6
syntax, A-12
time/event scheduling, A-2
UNIX codes, A-9
user variable, A-7
Event scheduling error messages, A-2
Events
and data transfers, 2-10
designing, 2-2
invalid triggers, B-1
triggering, 3-7
Explicit data type assignment, 2-8

F

File I/O error messages, A-8
Files
compile theory, 4-4
data transfer specification, 3-2
locations, 1-6, 5-2

G

General error messages, A-10
Glossary, 1-9
Grouping data transfer by event, 2-4

H

Hardware components, 1-4
Hexadecimal conversion, D-1

I

Implied data type assignment, 2-8
Installation
Oracle networking file, 1-11
RDT option, 1-10
RDT program, 4-11
RDT program to network primary, 4-17
testing RDT programs, 5-3
Instruction syntax, C-1
Interface components, 1-5
Internal error messages, A-11

K

Killing, RDT program, 4-13

L

Linking, data transfer with events, 2-2

M

Modifying. *See* Changing
Moving, rows of data, 3-4

O

Operation theory, 1-7

Oracle database format, installing networking file, 1-11

P

Planning, data transfer program, 2-9

Priority (UNIX), RDT program, 4-10

Program administration, 4-2

Programming hints, 3-4

R

Remote data transfer (RDT)
- associating events, 2-2, 2-10
  - in multiple-station systems, 2-11
- connection mode, 4-9
- connection retries, 4-10
- creating specification file, 2-12
- description, 1-2
- design guidelines, 2-9
- directory structure, 1-6, 5-2
- error classes, A-17
- group, defined, 2-4
- guidelines, 1-3
- hardware components, 1-4
- installing networking file, Oracle, 1-11
- installing option, 1-10
- interface components, 1-5
- log file, 5-2
- operation theory, 1-7
- prerequisites, 1-8
- program administration, 4-2
- programming
  - accessing a table, 3-5
  - compiling, 4-6
  - defining transfer items, 3-5
  - deleting, 4-15
  - hints, 3-4
  - installing, 4-11
  - modifying, 4-9
  - moving multiple tuples, 3-4
  - number of logins, 3-6
  - options, 4-9
  - order of tasks, 2-6
  - removing, 4-14
  - starting/halting, 4-12
  - triggering events, 3-7, B-1
  - updating, 4-16
  - using bit tests, 3-5
- runtime error messages, A-14
- software components, 1-4
- testing programs, 5-3
- trace lines, 4-9
- tracing operation, 5-10
- transferring files, 4-17
- UNIX priority, 4-10

Removing, RDT program, 4-14

Repeat event, 3-14

Runtime errors, debugging, 5-6

S

Software components, 1-4

Start event, 3-14

Starting/halting, RDT program, 4-12

Synchronization, transferring RDT files, 4-17

Syntax
- error messages, A-12
- instruction, C-1

T

Terminology, 1-9

Testing data transfer program, 5-4

Time scheduling error messages, A-2

Trace option, 5-10

Trace output file
- example, 5-10
- number of lines, 4-9

Tracing, RDT program execution, 5-10

Transfer log, 5-2

Transferring, RDT program, 4-17

Trigger event, RDT, 3-7, B-1

Troubleshooting
  communication loss, remote database, 5-7
  communications, 5-8
  compile and installation, 5-2
  compile errors, 4-7
  constant value errors, A-6
  data type errors, A-5
  database errors, A-4
  disk space (with Oracle), 1-13, 5-9
  error classes, A-17
  file I/O errors, A-8
  file system alarm (with Oracle), 5-9
  general error messages, A-10
  internal errors, A-11

Troubleshooting (continued)
  limited logins, 3-6
  program execution delay, 5-6
  runtime errors, 3-5
  runtime, data transfer errors, A-14
  syntax errors, A-12
  time/event scheduling errors, A-2
  UNIX error codes, A-9
  user variable errors, A-7
  using error log file, 5-6

U

UNIX error codes, A-9

Update, RDT program, 4-16

User variable error messages, A-7

Using bit tests, 3-5

Customer Response

We would like to know what you think about our user manuals so that we can serve you better. How would you rate the quality of our manuals?

	Excellent	Good	Fair	Poor
Accuracy	_____	_____	_____	_____
Organization	_____	_____	_____	_____
Clarity	_____	_____	_____	_____
Completeness	_____	_____	_____	_____
Graphics	_____	_____	_____	_____
Examples	_____	_____	_____	_____
Overall design	_____	_____	_____	_____
Size	_____	_____	_____	_____
Index	_____	_____	_____	_____

Would you be interested in giving us more detailed comments about our manuals?

☐ **Yes!** Please send me a questionnaire.

☐ **No.** Thanks anyway.

Your Name: _____

Title: _____

Telephone Number: (____) _____

Company Name: _____

Company Address: _____

_____

_____

Manual Name: SIMATIC PCS 7 OSx 4.1.2 Remote Data Transfer Manual **Edition:** Original
Manual Assembly Number: 2811154-0001 **Date:** 7/02
Order Number: 6ES7 6550XX058BD2

FOLD

SIEMENS ENERGY & AUTOMATION INC
3000 BILL GARLAND ROAD
P O BOX 1255
JOHNSON CITY TN 37605-1255

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO.3 JOHNSON CITY, TN

POSTAGE WILL BE PAID BY ADDRESSEE

ATTN TECHNICAL COMMUNICATIONS M/S 519
SIEMENS ENERGY & AUTOMATION INC
P O BOX 1255
JOHNSON CITY TN 37605-1255

FOLD