Rational Networking—TCP/IP Reference Manual

Telnet (TEL)

Document Control Number: 8003A-01 (803-002329)

Rev. 1.0, July 1986
Rev. 2.0, July 1987 (Delta)

Note the Reader's Comments form on the last page of this book, which requests the user's evaluation to assist Rational in preparing future documentation.

Rational
1501 Salado Drive
Mountain View, California 94043

Contents

RATIONAL

How to Use This Book

The Telnet (TEL) book of the *Rational Networking—TCP/IP Reference Manual* describes the Ada® package specifications for the facilities provided by the Rational Networking—TCP/IP implementation of the Telnet protocol. This book is intended for users who are familiar with the Rational Environment™ and with Ada programming.

Organization of the Networking Manual

The *Rational Networking—TCP/IP Reference Manual* (Networking Manual for brevity) includes the following volumes:

1	Telnet (TEL)
	File Transfer Protocol (FTP)
2	Transport Layer (TRL)
	Remote Procedure Call (RPC)

Each *volume* of the Networking Manual contains two *books* separated by large colored tabs. Each book contains information on particular features or areas of application in networking. The abbreviation for the name of each book (for example, TEL for Telnet) appears on the binder cover and spine, and this abbreviation is used in page numbers and cross-references. The books grouped into a given volume are logically related.

The Networking Manual provides reference information organized to efficiently answer specific questions about the Rational Networking product. Products other than the Networking product are documented in individual manuals (for example, the *Rational Environment Reference Manual* or the *Rational Target Build Utility Reference Manual*).

Volume 1

Volume 1 documents the commands used in the Rational Networking product. This volume contains the following two books:

- **Telnet**: The Telnet (TEL) book contains the commands used to establish and terminate connections to a remote host through a terminal logged into a local host.
- **File Transfer Protocol**: The File Transfer Protocol (FTP) book contains the commands used to transfer text and binary data files between two hosts. Packages File_Transfer, Ftp_Product, and Transfer_Generic can be used to develop programmatic interfaces for FTP.

Volume 1 also contains a Master Index that combines the index information from all four books of the Networking Manual.

Volume 2

Volume 2 documents the packages provided to develop new programmatic interfaces. This volume contains the following two books:

- **Transport Layer**: The Transport Layer (TRL) book describes the concepts and interfaces used to build networking tools for specific applications.
- **Remote Procedure Call**: The Remote Procedure Call (RPC) book describes the concepts and interfaces used to write clients and servers for remote procedure calls, in which an application running on one host can make procedure calls to applications running on different hosts.

Volume 2 also contains a Master Index that combines the index information from all four books of the Networking Manual.

Book Organization

Each book begins with a colored tab on which the name of the book appears. Each book typically contains the following sections:

- **Key Concepts**: This section describes the key concepts that pertain to the networking facilities documented in the book. The section is located behind its own tab following the How to Use This Book section.
- **Unit sections**: Each of the commands, tools, and so on has a declaration within an Ada compilation unit (typically a package). For each unit, there is a section that contains reference entries for the declarations within that unit. Each section is preceded by a tab.

  The sections for units are alphabetized by the simple names of the units. For example, the section for package !Tools.Network.Revn.Units.Network_Product is alphabetized under Network_Product.

  For many units, introductory material and/or examples specific to the unit appear after the section tabs.

  Within the section of a given unit, the reference entries describing the unit's declarations are organized alphabetically after the section introduction. Appearing at the top of each page of a reference entry are the simple name of the name of the given declaration and the fully qualified pathname of the enclosing unit.
- **Index**: Preceded by a tab, the Index appears as the last section of each book. It contains entries for each unit or declaration, along with additional topical

references. Each book index covers only the material documented in that particular book. Each volume contains a Master Index that provides entries for the information documented in all the books within the Networking Manual.

Italic page numbers indicate the page on which the primary reference entry for a declaration appears; nonitalic page numbers indicate key concepts, defined terms, cross-references, and exceptions raised.

Suggestions for Finding Information

The following suggestions can help you find various kinds of information in the documentation for Rational's products.

Learning about New Facilities

If you are a novice user starting to use the Rational Environment, consult the *Rational Environment User's Guide.*

If you are familiar with the Rational Environment but are interested in learning about certain networking commands, for example, you might start by reading the Key Concepts for the specific book, which describes important concepts and gives helpful examples.

It can also be useful to glance through the introductions provided for some of the units in the book. These introductions, located immediately after the tabs for the units, often contain helpful examples.

Finding Information on a Specific Item

If you know the name of the item and the book in which it is documented, consult either the table of contents or the index for that book. You can also turn through the pages of the book using the names and pathnames of the reference entries to locate the entry that you want. Remember that the reference entries for a unit are organized alphabetically by simple name within a tabbed section.

If you know the simple name of the entry but do not know the book in which it is documented, look in the Master Index to find the book abbreviation and page number.

If you cannot find an item in the Master Index, the item either is not documented or is documented in manuals for a product other than the Rational Networking—TCP/IP product (for example, the Rational Environment or the Rational Target Build Utility). If you know the pathname, consult the World ! section of the Reference Summary in Volume 1 of the *Rational Environment Reference Manual* to determine whether the item is documented and in which manual.

Using the Index

The index of each book contains entries for each unit and its declarations, organized alphabetically by simple name. When using the index to find a specific item, consult the italic page number for the primary reference to that item. Nonitalic page numbers indicate key concepts, defined terms, cross-references, and exceptions raised.

Viewing Specifications On-Line

If you know the pathname of a declaration and want to see its specification in a window of the Rational Environment, provide its pathname to the !Commands.Common.Definition procedure—for example, `Definition ("!Tools.Ftp.Revn.Units-.Commands.Ftp");`. If you know the simple name of the unit in which the declaration appears, in most cases you can use searchlist naming as a quick way of viewing the unit—for example, `Definition ("\Ftp");`.

Using On-Line Help

Most of the information contained in the reference entries for each unit is available through the on-line help facilities of the Rational Environment. Press the $\boxed{\text{Help on Help}}$ key or consult the *Rational Environment User's Guide* or the *Rational Environment Reference Manual*, EST, Help, for more information on using this on-line help facility.

Cross-Reference Conventions

The following conventions are used in cross-references to information:

- **Specific page/book**: For references to a specific place in a book, the book abbreviation is followed by the page number in the book (for example, RPC-23). If the book abbreviation is omitted, the current book is implied (for example, the page numbers in the table of contents for a book do not include the book prefix).
- **Declaration in same unit**: References to the documentation for a declaration in the same unit are indicated by the simple name of the desired declaration. For example, within the reference entry for the Ftp.Connect procedure, a reference to the Ftp.Disconnect procedure is "procedure Disconnect." Note that if the unit contains nested packages, references to nested declarations use qualified pathnames.
- **Declaration in different unit, same book**: References to the documentation for a declaration in another unit are indicated by the qualified pathname of the desired declaration. For example, within the reference entry for the Ftp.Connect procedure, a reference to the Ftp_Profile.Auto_Login function is "function Ftp_Profile.Auto_Login."
- **Declaration in different book**: References to the documentation for a declaration in another book are indicated by the addition of the abbreviation for that book. For example, within the reference entry for the Ftp.Connect procedure, a reference to the Transport.Connect procedure in the Transport Layer book would be "TRL, procedure Transport.Connect."

Feedback to Rational: Reader's Comments Form

Rational wants to make its documentation as useful and error-free as possible. Please provide us with feedback. The last page of each book contains a Reader's Comments form that you can use to send us comments or to report errors. You can also submit problem reports and make suggestions electronically by using the SIMS problem-reporting system. If you use SIMS to submit documentation comments, please indicate the manual name, book name, and page number.

RATIONAL

Key Concepts

The Rational Networking—TCP/IP product forms an intersystem interface for host/target development. Networking facilitates:

- Downloading Ada source code and target object code
- Interactive debugging
- Controlling target execution
- Updating target state
- Communicating between multiple R1000s
- Building distributed applications and test scaffolds
- Porting software between R1000s and other machines

The Telnet protocol establishes connections to remote hosts through a terminal logged into a local host. The Rational Environment user can execute tasks on a remote machine without having to end the current session on the local host.

A Telnet remote login behaves as if the user were logged in through a direct connection to the remote machine. Telnet supports connections running in both directions—between two R1000 hosts or between an R1000 host and another system. Once a Telnet connection is established, the user has full access to system utilities on the remote host.

A Telnet implementation consists of a client and a server. For a connection to be established, a remote system must have a Telnet server implemented on it. To initiate a Telnet session from its terminal, a local system must have a Telnet client implemented on it. Not all systems have both a client and a server. A server-only or client-only system can act only as a remote system or only as a local system, respectively. The Rational Networking—TCP/IP product contains both a server and a client. The server is not visible to the user, and it runs as a background process in the Rational Environment. This section discusses only the Telnet client.

Naming

Using the Telnet client, each Telnet connection is distinguished by a unique, three-part Telnet connection name consisting of:

- A username composed of the local user login name and the associated Environment session number.
- A machine name for the remote machine to which the user is connected.
- A Telnet session number.

A sample Telnet connection name is:

```
Qed.S_1.The_Vax.1
```

where:

- `Qed` and `S_1` (Environment session) form the username.
- `The_Vax` is the machine name.
- `1` is the Telnet session number.

This naming convention allows a user to be connected to several remote systems at once or to have multiple Telnet connections to one remote machine.

Commands

All Telnet commands are executed by the local computer. This means that once a connection is established, the user must return to the local Environment to establish another connection or to release the original connection after use. In addition to the commands for connecting and disconnecting connections, several Telnet commands help the user to escape to the local machine Environment without killing the session and connection.

The Telnet user interface consists of four basic user commands and their parameters (the parameters discussed below also exist as session switches):

- Telnet.Connect: Sets up a connection between the local and remote computers. This command's most useful parameters are:
  - Escape (string value): Specifies the escape sequence for escaping to the local Environment. The escape sequence can be any nonnull character string.
  - Escape_On_Break (Boolean value): Specifies, when set to true, the escape sequence to be the break signal. Escape_On_Break as well as the Escape sequence string can be set to true, offering two ways to escape a session.
  - Remote_Machine: Specifies the remote machine into which you log. This parameter also is used for Telnet.Disconnect.
- Telnet.Disconnect: Breaks the connection, ends the Telnet session, and releases the resources. This command can be used periodically to free unused Telnet connections.

- Telnet.Send_Break: Sends a break signal to the remote machine. On many systems, this is the usual escape sequence.
- Telnet.Show_Sessions: Displays a list of current Telnet connections. Even if the user has escaped from the remote machine and returned to the local Environment, the Telnet connection still exists until a Telnet.Disconnect command executes, disconnecting the connection. Telnet.Show_Sessions gives the user the username and session information required by Telnet.Disconnect.

Switches and Defaults

Many parameter values are virtually constant for a given user. For common applications, it is much easier to have the values of these parameters automatically set to the commonly used values rather than to specify parameter values each time the command is used. The Rational Environment supplies ***session switches*** to do this.

Switches are stored in files. The user can edit the switch values in these files by means of the switch file object editor. The !Commands.Switches.Edit_Session_Attributes command edits session switch files. Refer to the documentation on Session Switches in the Session and Job Management (SJM) book of the *Rational Environment Reference Manual* for further information on switches.

There is a switch to provide a value for most Telnet parameters. If the user does not give a Telnet command a parameter value in the Telnet Command window, Telnet takes the parameter value from the user's session switch file. These parameter values are available through package Telnet_Profile.

The Rational Environment supports the following switches for Telnet:

- Escape
- Escape_On_Break
- Remote_Machine

The actions of the switch parameters are described in the following example and in the reference entries in this book.

Example of Command Use

The following example shows a typical use of the Telnet commands starting with remote login to a VAX™ named The_Vax from a local R1000 named Clem. The user's name is Qed and the Environment session is S_1. The example ends by breaking the connection and releasing the connection resources.

1. Log into Clem and enter the Telnet.Connect command in a Command window.
   The Command window appears as follows:

```
begin
   Telnet.Connect
         (Remote_Machine => Telnet_Profile.Remote_Machine,  Session => 1,
          Escape => Telnet_Profile.Escape,
          Escape_On_Break => Telnet_Profile.Escape_On_Break,
          Terminal => System_Utilities.Terminal);
end;
```

2. Enter the following parameters:

Remote_Machine	Set to The_Vax.
Escape	Use the default value for this parameter, which is "]".
Escape_On_Break	Set to true.
Terminal	Use the default value for this parameter, which is the login from the current terminal.

3. Press [Promote] to set up the connection. A log file displayed on the screen shows the progress of the connection. Once the connection is established, the screen momentarily goes blank and then puts the user at the remote computer login prompt, in this case The_Vax. The terminal is switched from Rational to ANSI mode.
4. After working on the remote machine, return to the local machine and either disconnect this connection or set up an additional connection. To escape to the local Environment, press [F5]. The screen goes blank for a moment and you are returned to your original working context in the local Environment. That is, the Telnet session is escaped (not disconnected) and you are brought back to your original state in the local Environment on Clem.
5. To examine the state of your Telnet connections, enter the Telnet.Show_Sessions command in the local Environment. The Command window appears as follows:

```
begin
    Telnet.Show_Sessions  (User => Telnet.My_User_Name);
end;
```

6. Press [Promote].
   A list of all Telnet connections for Qed.S_1 is shown, as follows:

```
TELNET sessions for QED.S_1

    session          terminal          escape
==============================================
QED.S_1.THE_VAX.1     16              "]", break
```

If you had entered just Telnet.Show_Sessions ("Qed"), all Telnet connections for all of Qed's Environment sessions would have been shown.

7. Once you have finished working on the remote machine, it is best to log off the remote machine before disconnecting the Telnet connection. To disconnect, enter the Telnet.Disconnect command. For example:

   ```
   begin
   Telnet.Disconnect("The_Vax");
   end;
   ```

8. Press [Promote].

   A status message appears when the Telnet connection is disconnected.

9. Verify that the connection has been disconnected by executing the Telnet.Show_Sessions command. The connection should no longer show up in the Telnet.Show_Sessions table.

RATIONAL

package Telnet

Package Telnet is the user interface for managing Telnet connections. This package sets up connections between a local terminal and a remote computer and disconnects them when the user is finished. The package also transmits data over Telnet connections.

procedure Connect

```
procedure Connect
(Remote_Machine   : Machine_Name           := Telnet_Profile.Remote_Machine;
 Session          : Session_Number         := 1;
 Escape           : String                 := Telnet_Profile.Escape;
 Escape_On_Break  : Boolean                := Telnet_Profile.Escape_On_Break;
 Terminal         : System_Utilities.Port  := System_Utilities.Terminal);
```

Description

Establishes a new connection or resumes an existing connection with the specified computer (Remote_Machine) and session (Session).

Upon receipt of the specified escape sequence from the terminal, the connection is escaped and the terminal is reconnected to the local Environment.

Parameters

`Remote_Machine : Machine_Name := Telnet_Profile.Remote_Machine;`

Specifies the name of the remote machine to which to connect. The default is the name of the machine specified by the Remote_Machine switch in the session switch file.

`Session : Session_Number := 1;`

Specifies the number of the session to which to connect. Multiple connections to a single remote machine are distinguished by different session numbers. The default is 1.

`Escape : String := Telnet_Profile.Escape;`

Specifies the escape sequence for escaping this connection. The default is specified in the Escape switch in the session switch file.

When the escape sequence is sent by the terminal, the connection is suspended and the terminal is reconnected to the local Environment.

The escape sequence can be set to any string. If the null string ("") is specified, there will be no way to escape from the connection by typing data keys. If a nonnull string is specified, typing that string, with no more than a 1-second delay between characters, will escape from the connection.

```
Escape_On_Break : Boolean := Telnet_Profile.Escape_On_Break;
```

Specifies whether to accept break signals from the terminal to escape this connection. The default is specified in the Escape_On_Break switch in the session switch file.

If Escape_On_Break is set to true, and a break signal is sent by the terminal, the connection is escaped.

```
Terminal : System_Utilities.Port := System_Utilities.Terminal;
```

Specifies the local terminal from which to initiate a session. The default is the terminal into which the user is logged. This parameter is useful for testing.

procedure Disconnect

```
procedure Disconnect
         (Remote_Machine  : Machine_Name   := Telnet_Profile.Remote_Machine;
          Session         : Session_Number := 1;
          User            : User_Name      := Telnet.My_User_Name);
```

Description

Disconnects a connection with the specified computer (Remote_Machine) and session (Session).

Parameters

`Remote_Machine : Machine_Name := Telnet_Profile.Remote_Machine;`

Specifies the name of the remote machine from which to disconnect. The default is the value specified in the Remote_Machine switch in the session switch file.

`Session : Session_Number := 1;`

Specifies the number of the session to be disconnected. The default is 1.

`User : User_Name := Telnet.My_User_Name;`

Specifies the local username and local session name for the connection to be disconnected. The default is the caller's username and session name.

subtype Machine_Name

```
subtype Machine_Name is String;
```

Description

Specifies the name of a remote machine.

function My_User_Name

```
function My_User_Name return User_Name;
```

Description

Returns your own username and session name.

Parameters

```
return User_Name;
```

Returns the local username and local session name of the caller.

procedure Send

```
procedure Send
        (Data           : String         := Telnet_Profile.Escape;
         Remote_Machine : Machine_Name   := Telnet_Profile.Remote_Machine;
         Session        : Session_Number := 1);
```

Description

Sends data via an existing Telnet connection.

Parameters

`Data :  String := Telnet_Profile.Escape;`

Specifies the data to be sent.

`Remote_Machine :  Machine_Name := Telnet_Profile.Remote_Machine;`

Specifies the name of the remote machine to which to send data.

`Session :  Session_Number := 1;`

Specifies the session number of the connection over which to transmit data.

procedure Send_Break

```
procedure Send_Break
        (Remote_Machine : Machine_Name   := Telnet_Profile.Remote_Machine;
         Session        : Session_Number := 1);
```

Description

Sends a break signal via an existing Telnet connection.

Parameters

`Remote_Machine : Machine_Name := Telnet_Profile.Remote_Machine;`

Specifies the name of the remote machine to which to send a break signal.

`Session : Session_Number := 1;`

Specifies the session number of the connection through which to send a break signal.

subtype Session_Number

```
subtype Session_Number is Positive;
```

Description

Specifies the session number of a Telnet connection.

The default session number is 1. Specifying different session numbers makes it possible to have multiple connections to a remote machine.

procedure Show_Sessions

```
procedure Show_Sessions (User : User_Name := Telnet.My_User_Name);
```

Description

Displays all existing connections for the specified user.

Fields shown include local username, local session name, remote machine name, and Telnet session number.

Parameters

```
User : User_Name := Telnet.My_User_Name;
```

Specifies the user (and session) for whom to display connections. By default, this parameter displays the connections for the current user. The format is:

user's local login name.local Environment session name

The "?" parameter value produces a listing for all users.

subtype User_Name

```
subtype User_Name is String;
```

Description

Specifies the username for a Telnet connection.

The format for the username is:

user's local login name.local Environment session name

end Telnet;

RATIONAL

package Telnet_Profile

Package Telnet_Profile supplies default values for Telnet commands. Each of these functions has a corresponding session switch that can be set by the user.

Many parameter values are virtually constant for a given user. For common applications, it is much easier to have the values of these parameters automatically set to the commonly used values rather than to specify parameter values each time the command is used. The Rational Environment supplies session switches to do this. Session switches set parameter values at login. Telnet uses these session values for the duration of the login session.

Switches are stored in files. The user can edit the switch values in these files by means of the switch file object editor. The !Commands.Switches.Edit_Session_Attributes command edits session switch files. Refer to the documentation on Session Switches in the Session and Job Management (SJM) book of the *Rational Environment Reference Manual* for further information on switches.

There is a switch to provide a value for most Telnet parameters. If the user does not give a Telnet command a parameter value in the Telnet Command window, Telnet takes the parameter value from the user's session switch file.

function Escape

```
function Escape return String;
```

Description

Returns the current value of the Escape session switch.

Parameters

```
return String;
```

Returns the value of the Escape session switch. The default is "]".

References

procedure Telnet.Connect

function Escape_On_Break

```
function Escape_On_Break return Boolean;
```

Description

Returns the current value of the Escape_On_Break session switch.

Parameters

```
return Boolean;
```

Returns the value of the Escape_On_Break session switch. The default is true.

References

procedure Telnet.Connect

function Remote_Machine

```
function Remote_Machine return String;
```

Description

Returns the current value of the Remote_Machine session switch.

Parameters

```
return String;
```

Returns the value of the Remote_Machine session switch. The default value is the null string ("").

References

procedure Telnet.Connect

procedure Telnet.Disconnect

procedure Telnet.Send

procedure Telnet.Send_Break

end Telnet_Profile;

Index

This index contains entries for each unit and its declarations as well as definitions, topical cross-references, exceptions raised, errors, enumerations, pragmas, switches, and the like. The entries for each unit are arranged alphabetically by simple name. An italic page number indicates the primary reference for an entry.

RATIONAL

RATIONAL

READER'S COMMENTS

Note: This form is for documentation comments only. You can also submit problem reports and comments electronically by using the SIMS problem-reporting system. If you use SIMS to submit documentation comments, please indicate the manual name, book name, and page number.

Did you find this book understandable, usable, and well organized? Please comment and list any suggestions for improvement.

____
____
____
____
____
____

If you found errors in this book, please specify the error and the page number. If you prefer, attach a photocopy with the error marked.

____
____
____
____

Indicate any additions or changes you would like to see in the index.

____
____
____
____

How much experience have you had with the Rational Environment?

6 months or less _______ 1 year _______ 3 years or more ______

How much experience have you had with the Ada programming language?

6 months or less _______ 1 year _______ 3 years or more ______

Name (optional)______________________ Date______________
Company ______________________________
Address ______________________________
City ____________________ State ____________ ZIP Code ____________

Please return this form to:

Publications Department
Rational
1501 Salado Drive
Mountain View, CA 94043

Rational Networking—TCP/IP Reference Manual

File Transfer Protocol (FTP)

Document Control Number: 8003A-01 (803-002330)

Rev. 1.0, November 1985
Rev. 2.0, July 1986
Rev. 3.0, July 1987 (Delta)

Note the Reader's Comments form on the last page of this book, which requests the user's evaluation to assist Rational in preparing future documentation.

Rational
1501 Salado Drive
Mountain View, California 94043

Contents

How to Use This Book

The File Transfer Protocol (FTP) book of the *Rational Networking—TCP/IP Reference Manual* describes the Ada® package specifications for the facilities provided by the Rational Networking—TCP/IP implementation of the FTP protocol. This book is intended for users who are familiar with the Rational Environment™ and with Ada programming.

Organization of the Networking Manual

The *Rational Networking—TCP/IP Reference Manual* (Networking Manual for brevity) includes the following volumes:

1	Telnet (TEL)
	File Transfer Protocol (FTP)
2	Transport Layer (TRL)
	Remote Procedure Call (RPC)

Each *volume* of the Networking Manual contains two *books* separated by large colored tabs. Each book contains information on particular features or areas of application in networking. The abbreviation for the name of each book (for example, TEL for Telnet) appears on the binder cover and spine, and this abbreviation is used in page numbers and cross-references. The books grouped into a given volume are logically related.

The Networking Manual provides reference information organized to efficiently answer specific questions about the Rational Networking product. Products other than the Networking product are documented in individual manuals (for example, the *Rational Environment Reference Manual* or the *Rational Target Build Utility Reference Manual*).

Volume 1

Volume 1 documents the commands used in the Rational Networking product. This volume contains the following two books:

- **Telnet**: The Telnet (TEL) book contains the commands used to establish and terminate connections to a remote host through a terminal logged into a local host.
- **File Transfer Protocol**: The File Transfer Protocol (FTP) book contains the commands used to transfer text and binary data files between two hosts. Packages File_Transfer, Ftp_Product, and Transfer_Generic can be used to develop programmatic interfaces for FTP.

Volume 1 also contains a Master Index that combines the index information from all four books of the Networking Manual.

Volume 2

Volume 2 documents the packages provided to develop new programmatic interfaces. This volume contains the following two books:

- **Transport Layer**: The Transport Layer (TRL) book describes the concepts and interfaces used to build networking tools for specific applications.
- **Remote Procedure Call**: The Remote Procedure Call (RPC) book describes the concepts and interfaces used to write clients and servers for remote procedure calls, in which an application running on one host can make procedure calls to applications running on different hosts.

Volume 2 also contains a Master Index that combines the index information from all four books of the Networking Manual.

Book Organization

Each book begins with a colored tab on which the name of the book appears. Each book typically contains the following sections:

- **Key Concepts**: This section describes the key concepts that pertain to the networking facilities documented in the book. The section is located behind its own tab following the How to Use This Book section.
- **Unit sections**: Each of the commands, tools, and so on has a declaration within an Ada compilation unit (typically a package). For each unit, there is a section that contains reference entries for the declarations within that unit. Each section is preceded by a tab.

  The sections for units are alphabetized by the simple names of the units. For example, the section for package !Tools.Network.Revn.Units.Network_Product is alphabetized under Network_Product.

  For many units, introductory material and/or examples specific to the unit appear after the section tabs.

  Within the section of a given unit, the reference entries describing the unit's declarations are organized alphabetically after the section introduction. Appearing at the top of each page of a reference entry are the simple name of the name of the given declaration and the fully qualified pathname of the enclosing unit.
- **Index**: Preceded by a tab, the Index appears as the last section of each book. It contains entries for each unit or declaration, along with additional topical

references. Each book index covers only the material documented in that particular book. Each volume contains a Master Index that provides entries for the information documented in all the books within the Networking Manual.

Italic page numbers indicate the page on which the primary reference entry for a declaration appears; nonitalic page numbers indicate key concepts, defined terms, cross-references, and exceptions raised.

Suggestions for Finding Information

The following suggestions can help you find various kinds of information in the documentation for Rational's products.

Learning about New Facilities

If you are a novice user starting to use the Rational Environment, consult the *Rational Environment User's Guide.*

If you are familiar with the Rational Environment but are interested in learning about certain networking commands, for example, you might start by reading the Key Concepts for the specific book, which describes important concepts and gives helpful examples.

It can also be useful to glance through the introductions provided for some of the units in the book. These introductions, located immediately after the tabs for the units, often contain helpful examples.

Finding Information on a Specific Item

If you know the name of the item and the book in which it is documented, consult either the table of contents or the index for that book. You can also turn through the pages of the book using the names and pathnames of the reference entries to locate the entry that you want. Remember that the reference entries for a unit are organized alphabetically by simple name within a tabbed section.

If you know the simple name of the entry but do not know the book in which it is documented, look in the Master Index to find the book abbreviation and page number.

If you cannot find an item in the Master Index, the item either is not documented or is documented in manuals for a product other than the Rational Networking—TCP/IP product (for example, the Rational Environment or the Rational Target Build Utility). If you know the pathname, consult the World ! section of the Reference Summary in Volume 1 of the *Rational Environment Reference Manual* to determine whether the item is documented and in which manual.

Using the Index

The index of each book contains entries for each unit and its declarations, organized alphabetically by simple name. When using the index to find a specific item, consult the italic page number for the primary reference to that item. Nonitalic page numbers indicate key concepts, defined terms, cross-references, and exceptions raised.

Viewing Specifications On-Line

If you know the pathname of a declaration and want to see its specification in a window of the Rational Environment, provide its pathname to the !Commands.Common.Definition procedure—for example, `Definition ("!Tools.Ftp.Revn.Units-.Commands.Ftp");`. If you know the simple name of the unit in which the declaration appears, in most cases you can use searchlist naming as a quick way of viewing the unit—for example, `Definition ("\Ftp");`.

Using On-Line Help

Most of the information contained in the reference entries for each unit is available through the on-line help facilities of the Rational Environment. Press the [Help on Help] key or consult the *Rational Environment User's Guide* or the *Rational Environment Reference Manual*, EST, Help, for more information on using this on-line help facility.

Cross-Reference Conventions

The following conventions are used in cross-references to information:

- **Specific page/book**: For references to a specific place in a book, the book abbreviation is followed by the page number in the book (for example, RPC-23). If the book abbreviation is omitted, the current book is implied (for example, the page numbers in the table of contents for a book do not include the book prefix).
- **Declaration in same unit**: References to the documentation for a declaration in the same unit are indicated by the simple name of the desired declaration. For example, within the reference entry for the Ftp.Connect procedure, a reference to the Ftp.Disconnect procedure is "procedure Disconnect." Note that if the unit contains nested packages, references to nested declarations use qualified pathnames.
- **Declaration in different unit, same book**: References to the documentation for a declaration in another unit are indicated by the qualified pathname of the desired declaration. For example, within the reference entry for the Ftp.Connect procedure, a reference to the Ftp_Profile.Auto_Login function is "function Ftp_Profile.Auto_Login."
- **Declaration in different book**: References to the documentation for a declaration in another book are indicated by the addition of the abbreviation for that book. For example, within the reference entry for the Ftp.Connect procedure, a reference to the Transport.Connect procedure in the Transport Layer book would be "TRL, procedure Transport.Connect."

Feedback to Rational: Reader's Comments Form

Rational wants to make its documentation as useful and error-free as possible. Please provide us with feedback. The last page of each book contains a Reader's Comments form that you can use to send us comments or to report errors. You can also submit problem reports and make suggestions electronically by using the SIMS problem-reporting system. If you use SIMS to submit documentation comments, please indicate the manual name, book name, and page number.

RATIONAL

Key Concepts

The Rational Networking—TCP/IP product forms an intersystem interface for host/target development. Networking facilitates:

- Downloading Ada source code and target object code
- Interactive debugging
- Controlling target execution
- Updating target state
- Building test scaffolds
- Distributed processing

File Transfer Protocol (FTP) is used to transfer text files and binary data files between two R1000 hosts or between an R1000 host and another system. Users can store their files on a non-Rational host and use the foreign host's tools.

FTP provides a simple and flexible bidirectional file transfer mechanism between two or more machines. It offers both a command interface and a programmatic interface. This section primarily describes use of the command interface.

The Ftp, Ftp_Defs, Ftp_Name_Map, and Ftp_Profile packages are for the FTP user who is interested in a simple, automatic user interface for file transfers. This book discusses:

- Various methods for transferring data and text files
- Transferring files between R1000s and between an R1000 and a non-R1000 system
- Preserving library structures during a transfer
- Using macro commands for rapid execution of common operations
- Using switches to automatically supply commonly used parameter values

Packages File_Transfer, Ftp_Product, and Transfer_Generic are for more advanced users who want to use the programmatic interfaces available for FTP.

File Transfer Applications

File transfers occur between a local and a remote machine. A transfer can send a file either:

- From the local machine to the remote machine (Store or Put)
- From the remote machine to the local machine (Retrieve or Get)

During all FTP operations, the user is informed of the progress of the file transfer by a screen window displaying status and diagnostic messages as a log file.

It should be noted that FTP cannot transfer Ada units. FTP treats all data objects as files. Already compiled Ada units are transferred as text files; they can be recompiled when they reach their destination. Users who want to transfer compiled Ada units between R1000s should refer to the !Commands.Archive.Copy procedure, documented in the Library Management (LM) book of the *Rational Environment Reference Manual.*

Servers and Clients

During a data transfer session, the user logs into a local machine and then invokes a program called a *client.* The client uses the Transport Layer to set up a remote connection and logs in through this command connection to a remote machine. Throughout the dialogue, the client uses the command connection to send all user requests and query responses (passwords and the like) to the remote machine and to receive status messages from the remote machine. In short, the client acts as the active initiator of the dialogue.

Once a data transfer has been requested, the remote machine calls up a program, called a *server,* that has been passively waiting for the client's service request. The server handles:

- Searching for the remote file and transmitting it to the local machine
- Receiving a file from the local client and storing it in its remote destination file

Once the command connection is set up and a request for a file transfer is made, the server, using the Transport Layer, responds by setting up a data transfer connection back to the local machine over which files are sent.

For the most part, the server is invisible to the user, because all user interaction proceeds through the client to the server.

Sessions and Logging In

FTP is based on the idea of a transfer *session*. A session consists of:

- Establishing a connection
- Logging into the remote machine
- Transferring files
- Disconnecting

A session also has a certain state set up by the command parameter specifications. The session state consists of all the various parameter values that characterize it:

- Transfer type
- Remote operating system
- Current working remote directory

As will be seen in "Macro Transfer Commands," below, many of these operations can be automatically executed by the client through higher-level commands.

Transfer Types

Before transferring a file, the user needs to specify several parameters about the representation and structure of the file during the transfer. FTP supports the following transfer types:

- Text (ASCII)
- Image
- Binary

Text Transfer

Text transfer is used to send text files between systems. Text files contain special control characters such as carriage return and end-of-line and end-of-file characters that are represented differently on an R1000 and a non-R1000 system. A text transfer converts the control characters on one system into those appropriate for another system. These conversions introduce overhead into the transfer operation. Thus, the text transfer is the least efficient of the transfer types and should be used only to send text files between dissimilar machines. Refer to the Ftp_Defs.Type_Code literals for parameter values.

Image Transfer

An image transfer sends a byte-aligned file between machines as a stream of bytes. Because an image transfer does no conversion of control characters, it is used either for sending a nontext file between an R1000 and a non-R1000 system or for sending a text file between R1000s.

Binary Transfer

A binary transfer sends a nonbyte-aligned, arbitrary bit-length file as a stream of bits, preserving length and bit alignment. Padding bits are placed at the end of the file to reach a byte boundary. An extra byte is then added at the end of the file telling how many padding bits were inserted into the file. Upon arrival, this padding is stripped out to restore the original file. A binary transfer can be used only for sending a file between R1000s.

Name Mapping

Different systems use different filenaming conventions. FTP supplies default filename conversions when transferring files between systems. FTP currently supplies name conversions for UNIX®, AOS, VMS™, and the R1000. The conventions are:

- UNIX = /directory/subdirectory/filename
- AOS = :directory:subdirectory:filename.filetype
- VMS = [directory.subdirectory]filename.filetype
- R1000 = !directory.subdirectory.filename

Refer to the appropriate vendor documentation for naming conventions.

Note that you can use package Transfer_Generic to implement other name mappings.

Transferring Multiple Files

Single command transfers of related files can be accomplished by either:

- Transferring all the files in a directory
- Exploiting similarities in filenames through the use of wildcards

All the FTP commands that end in _Set support multiple file transfers and wildcards.

Three examples of file transfer are given at the end of this section. The first shows the transfer of a single file, the second shows the transfer of a group of files within a single directory, and the third shows the transfer of a series of dispersed files sharing a name fragment. Refer to these examples to see how to perform these types of transfers.

Transferring Sets of Files

The FTP _Set commands accept wildcards and set-naming conventions in their filename specifications. The _Set commands allow the user to transfer a set of related files scattered throughout multiple directories.

Files grouped together in a directory, as well as in separate directories, can be transferred using wildcards. The wildcard or set-naming conventions used must be those of the system from which the file is taken. For example, a Put_Set command always uses the R1000's naming conventions because the file is always originating on an R1000. A Get_Set command taking a file from a VAX™/VMS™ system, for example, and sending it to an R1000 uses the VMS conventions.

For more information on R1000 filenaming conventions, refer to the Library Management (LM) book of the *Rational Environment Reference Manual.*

Transferring Files in Multiple Directories

Establishing correspondences between data structures on dissimilar machines is essential for preserving the organization of those structures. FTP sets up mappings between directory and file structures on the remote machine and those on the local machine. These mappings ensure that file hierarchy is preserved and naming conventions are compatible when files are transferred between machines. These mappings are built around the concept of a *roof*, which is an ancestor directory of the files to be moved. Specifying a roof gives FTP a location for the files and logically groups those files together.

By specifying both a remote and a local roof, a transfer of the nested subdirectories will preserve their hierarchical organization when the files are copied. This type of transfer is called an *isomorphic* transfer (see Figure 1-1).

It is also possible to take a tree directory structure and transfer its files into a single leveled directory, thus "flattening out" the structure. This type of transfer is called a *nonisomorphic* transfer (Figure 1-2). Specifying the null ("") local roof as the source roof causes a nonisomorphic transfer. Clearly, a nonisomorphic transfer is not reversible.

Multiple-Directory Transfers Using Set-Naming Conventions

During a transfer using both roof directories and wildcards, FTP tries to create as close a match as possible between the original source directory structure and the new destination directory structure. Rules that govern this match are:

- A local file will be transferred isomorphically only if the subdirectories containing it already exist as subdirectories of the remote roof.
- If a subdirectory containing a local file to be transferred does not already reside under the remote roof, the transfer of the enclosed local file will fail.
- If the local roof is not an ancestor directory of a file to be transferred, the file will be transferred nonisomorphically ("flattened out"). In this case, the target file will be immediately within the directory named by the target roof.

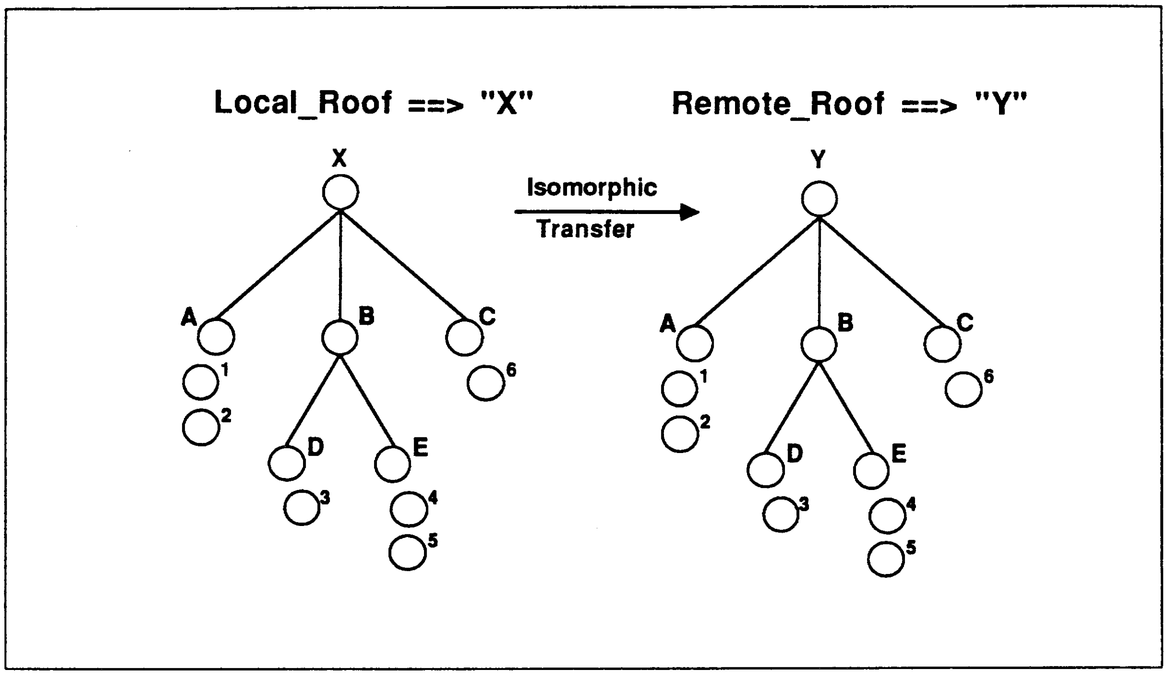

Figure 1-1. Isomorphic Transfers

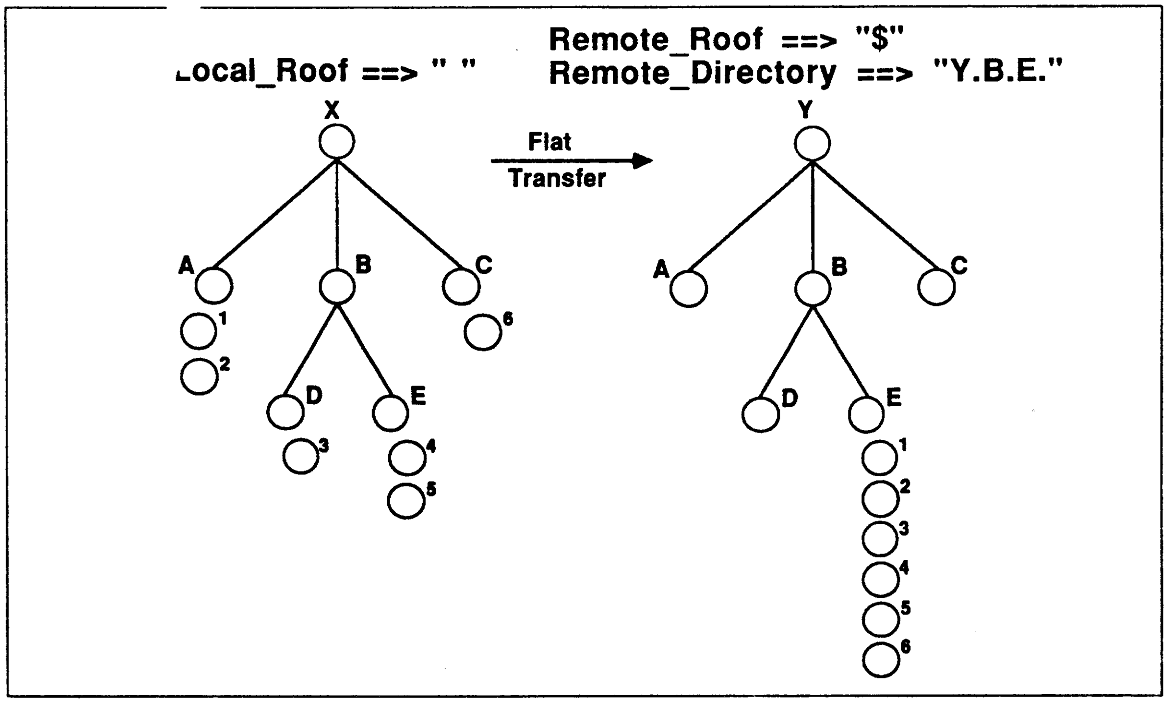

Figure 1-2. Nonisomorphic Transfers

For an example of these rules, refer to the transfer shown in Figure 1-3. Illustrated is the transfer between two R1000s (hence the use of R1000 wildcards and filenames) of all files specified by "!Users.Fred.?.Junk@". The local roof has been specified as "!Users.Fred.My_Stuff_1". The remote roof has been specified as "!Users.Fred.My-_Stuff_New". Notice that:

- The transfer of "Junk_1" and "Junk_2" illustrates the first rule.
- The failure to transfer "Junk_3" and "Junk_4" illustrates the second rule.
- The transfer of "Junk_5" and "Junk_6" illustrates the third rule.

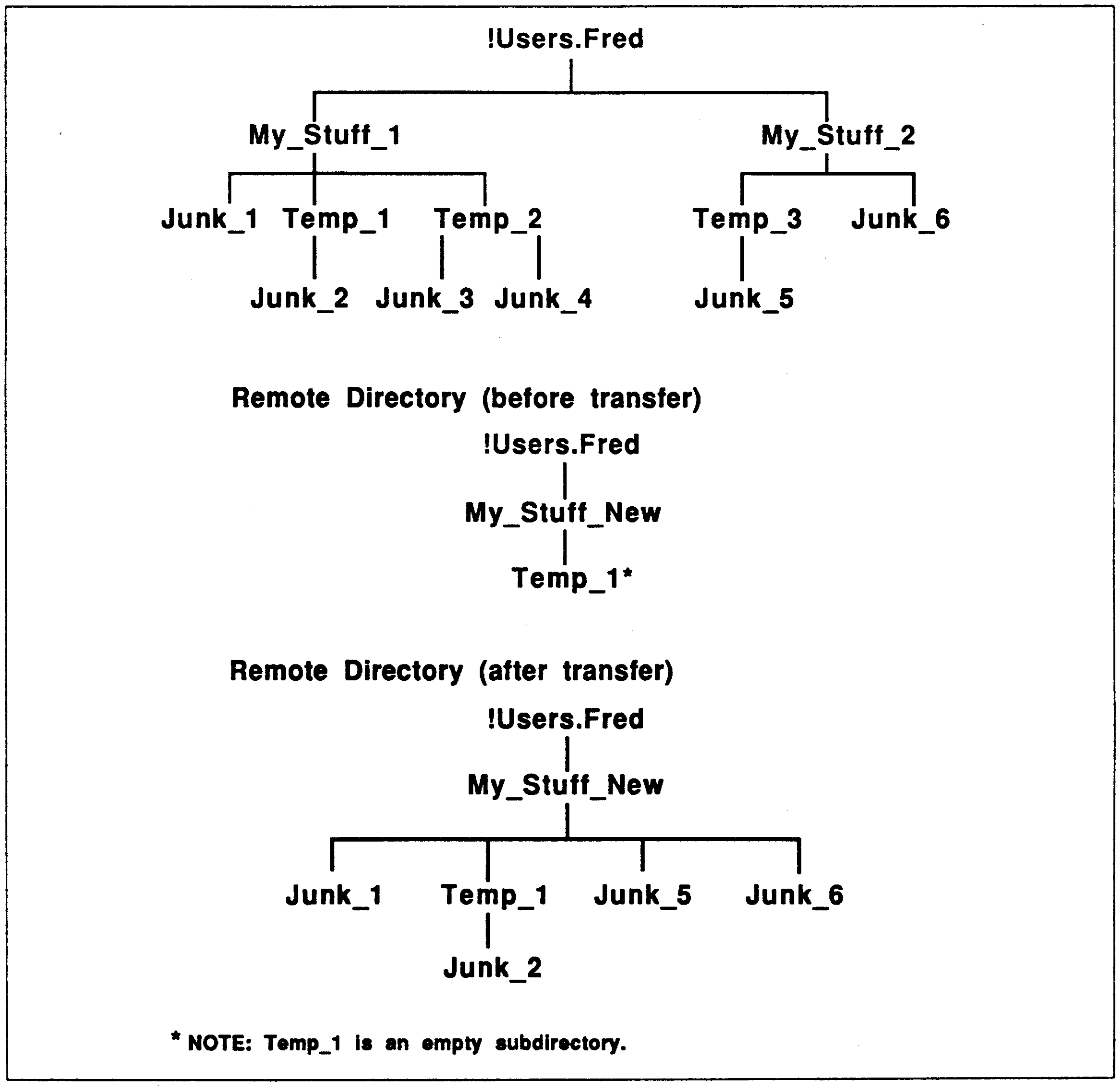

Figure 1-3. Directory Transfer Using Wildcards

Switches and Defaults

Many parameter values are virtually constant for a given user. For common applications, it is much easier to have the values of these parameters automatically set to the commonly used values rather than to specify parameter values each time the command is used. The Rational Environment supplies two types of switches to do this:

- The *session switches* set parameter values at login. FTP uses these session values for the duration of the login session.
- The *library switches* set parameter values for a given library. For transfers done from a Command window inside a given library, FTP uses the switch values associated with that library.

Switches are stored in files. The user can edit the switch values in these files by using the editing operations that apply to switch files. The !Commands.Switches.Edit command edits library switch files. The !Commands.Switches.Edit_Session_Attributes command edits session switch files. Refer to the Session and Job Management (SJM) and Library Management (LM) books of the *Rational Environment Reference Manual* for further information on switches.

There is a switch to provide a value for most FTP parameters. If the user does not give the file transfer command a parameter value in the FTP Command window:

1. FTP tries to get a value from the appropriate library switch file.
2. If no value has been explicitly set for the library switch, FTP tries to take the parameter value from the user's session switch file.
3. If no value has been set for the session switch, FTP tries to take the parameter value from one of the following files:
   - Remote password file for username or password
   - Remote session file for session names
4. If no value has been set in the remote files, FTP takes the standard default.

By keeping this search sequence in mind, the user can set a combination of library, session, and standard values for FTP's parameter values. These values are available through package Ftp_Profile.

The Rational Environment supports the following switches for FTP:

Account	Auto_Login
Password	Prompt_For_Account
Prompt_For_Password	Remote_Directory
Remote_Machine	Remote_Roof
Remote_Type	Send_Port_Enabled
Transfer_Mode	Transfer_Structure
Transfer_Type	Username

The actions of these parameter values are described in the examples at the end of this section and in the reference entries in this book.

Macro Transfer Commands

As you become familiar with the FTP commands, you will notice that many of the higher-level commands are merely combinations of more basic commands. For example, the Ftp.Put procedure:

- Establishes a connection to the remote machine
- Logs into that machine
- Locates the desired file on the local machine
- Transfers a copy of the file to the remote machine
- Breaks the connection

This single procedure combines these more basic procedures:

- Ftp.Connect
- Ftp.Login
- Ftp.Store
- Ftp.Disconnect

into a single command. Using a macro command like Ftp.Put may be more useful for ordinary tasks than a series of individual commands.

For programs automatically transferring multiple files, macro commands are lower in performance than the sum of their individual commands. With macros, a connection has to be reestablished for the transfer of each file rather than several transfer operations occurring on a single connection. Another problem with macros is that a mistaken entry in the command line will kill the entire command chain, not just the portion the command deals with directly. However, there are benefits to using macros:

- You do not need to "permanently" use up connections.
- You do not need to worry about releasing connections.

Some idea of the practical relationships between commands can be seen in the command list at the end of this section.

Examples of File Transfer

These examples serve to:

- Illustrate the basic user interface for FTP commands
- Show the use of macro commands
- Provide examples of two of the most frequently used FTP commands: Ftp.Put and Ftp.Put_Set

Transferring a Single File

This example illustrates the transfer of a single file using the Ftp.Put command. This type of transfer and its inverse, Ftp.Get, are the most common transfers used. This example assumes that the user is already logged into a local R1000. Some user-specific parameter values such as machine names and passwords are of String type and must be enclosed in double quotes. Other values for which there is a defined set of enumerated types (for example, Transfer_Type and Remote_Type) can be selected only from that set and are unquoted. Remember to give the package for *_Type constants, unless your Command window has a *use* clause set up to resolve unqualified names. In this example, *_Type constants must be preceded by "ftp."

This hypothetical example transfers a local R1000 text file named "!Users.Fred.Junk-_1" to a remote Data General computer named "Ed" running the AOS operating system and renames the file ":Udd:Fred:Junk_2".

To perform this Ftp.Put transfer:

1. Log into an R1000 as "fred" and enter the Ftp.Put command in a Command window or as part of a larger program.

   The Command window appears as follows:

```
begin
   Ftp.Put (From_Local_File => "", To_Remote_File => "",
            Remote_Machine => Ftp_Profile.Remote_Machine,
            Username => Ftp_Profile.Username,
            Password => Ftp_Profile.Password,
            Account => Ftp_Profile.Account,
            Remote_Directory => Ftp_Profile.Remote_Directory,
            Remote_Type => Ftp_Profile.Remote_Type, Append_To_File => False,
            Transfer_Type => Ftp_Profile.Transfer_Type,
            Transfer_Mode => Ftp_Profile.Transfer_Mode,
            Transfer_Structure => Ftp_Profile.Transfer_Structure,
            Send_Port => Ftp_Profile.Send_Port_Enabled,
            Response => Profile.Get),
end;
```

2. Enter the values for the following parameters:

`From_Local_File`	Supply the local pathname of the file to be moved. This pathname is relative to the currently prefixed local directory. In this example, the local file is named "junk_1".
`To_Remote_File`	Supply the pathname of the remote file to be created. This pathname is relative to the directory to be specified in the Remote_Directory parameter. In this example, the remote file is named "junk_2" and will reside in the user's remote home directory.
`Remote_Machine`	Supply the name of the remote machine. In this example, the remote machine is named "ed".
`Username`	Supply your remote machine username. In this example, the username is "fred".

Password	Supply your remote machine password. In this example, the password is "sly_guy".
Account	Supply your remote machine account, if any. No entry is given in this example, so FTP supplies a default value.
Remote_Directory	Supply the roof directory for new remote file. No entry is given in this example, so FTP supplies the user's remote home directory as a default.
Remote_Type	Supply the remote machine type (Ftp.Rational, Ftp.Aos, Ftp.Vms, Ftp.Unix, Ftp.Mvs). In this example, the machine type is Ftp.Aos.
Append_To_File	Set to true to append the file to a preexisting file of the same name. Set to false to overwrite a preexisting remote file of the same name. Pass over this parameter and the R1000 will supply the default value (false).
Transfer_Type	Supply the desired transfer type (which is taken from type Ftp_Defs.Transfer_Type). This is a transfer of a text file between an R1000 and a non-R1000, so the default value, text transfer (Ftp.Ascii), is supplied by the R1000.
Transfer_Mode	If you do not supply a value for this parameter, the R1000 will supply the default value.
Transfer_Structure	If you do not supply a value for this parameter, the R1000 will supply the default value.
Send_Port	If you do not supply a value for this parameter, the R1000 will supply the default value.
Response	If you do not supply a value for this parameter, the R1000 will supply the default value.

After the values are entered, the Command window appears as follows:

```
begin
   Ftp.Put (From_Local_File => "junk_1", To_Remote_File => "junk_2",
            Remote_Machine => "ed",
            Username => "fred",
            Password => "sly_guy",
            Account => Ftp_Profile.Account,
            Remote_Directory => Ftp_Profile.Remote_Directory,
            Remote_Type => Ftp.Aos, Append_To_File => False,
            Transfer_Type => Ftp.Ascii,
            Transfer_Mode => Ftp_Profile.Transfer_Mode,
            Transfer_Structure => Ftp_Profile.Transfer_Structure,
            Send_Port => Ftp_Profile.Send_Port_Enabled,
            Response => Profile.Get),
end;
```

3. Execute the command.
4. Note the messages that appear in the log file, which may be the current output window. These messages indicate when the transfer has completed or, if it has not completed successfully, why the transfer has failed.

Transferring Sets of Multiple Files Isomorphically

Multiple file transfers are accomplished using the _Set version of the single-file transfer commands Ftp.Put, Ftp.Get, Ftp.Store, and Ftp.Retrieve. The syntax and user interfaces for the multiple-file transfer commands behave like their single-file counterparts. However, the _Set commands do have a few different parameters that allow them to accept wildcards and directories.

This example transfers the contents of a local R1000 library named "!Users.Fred-.My_Stuff" to a remote MV computer named "Ed" and places the files in directory ":Udd:Fred:My_Stuff_New". Contained in this directory are text files and subdirectories of still more text files. Because this is an isomorphic transfer and it is assumed that the local directories from which the files are taken have counterparts on the remote machine, the directory relationships between files are preserved.

1. Bring up the Ftp.Put_Set command as you did previously. The Command window appears as follows:

```
begin
   Ftp.Put_Set (From_Local_File_Set => "", Local_Roof => "$",
                Remote_Roof => Ftp_Profile.Remote_Roof
                Remote_Machine => Ftp_Profile.Remote_Machine,
                Username => Ftp_Profile.Username,
                Password => Ftp_Profile.Password,
                Account => Ftp_Profile.Account,
                Remote_Directory => Ftp_Profile.Remote_Directory,
                Remote_Type => Ftp_Profile.Remote_Type,
                Append_To_File => False,
                Transfer_Type => Ftp_Profile.Transfer_Type,
                Transfer_Mode => Ftp_Profile.Transfer_Mode,
                Transfer_Structure => Ftp_Profile.Transfer_Structure,
                Send_Port => Ftp_Profile.Send_Port_Enabled,
                Response => Profile.Get),
end;
```

2. Enter the parameter values as before, except for the following:

From_Local_File_Set	Supply the local pathname of the files to be moved. Use wildcards to specify multiple files. This pathname is relative to the current local library. In this example, the local files are all contained in the library "!users.fred.my_stuff". Using wildcards, these files are specified as "!users.fred-.my_stuff.?".
Local_Roof	Supply the pathname of the local roof directory for the local file set. The default value "$" stands for the current directory. In this example, the directory is "!users.fred-.my_stuff".
Remote_Roof	Supply the pathname of the remote roof directory into which the files are to be moved. In this example, the directory is ":udd:fred:my_stuff_new".

`Remote_Directory`	Supply the remote directory for the connection to use as a prefix. A fully qualified remote roof pathname was specified for this transfer, so no entry is needed for Remote_Directory. FTP supplies the user's remote home directory as a default.

After the values are entered, the Command window appears as follows:

```
begin
 Ftp.Put_Set (From_Local_File_Set => "!users.fred.my_stuff?",
              Local_Roof => "!users.fred.my_stuff",
              Remote_Roof => ":udd:fred:my_stuff_new",
              Remote_Machine => "ed",
              Username => "fred",
              Password => "sly_guy",
              Account => Ftp_Profile.Account,
              Remote_Directory => Ftp_Profile.Remote_Directory,
              Remote_Type => Ftp_Profile.Remote_Type,
              Append_To_File => False,
              Transfer_Type => Ftp_Profile.Transfer_Type,
              Transfer_Mode => Ftp_Profile.Transfer_Mode,
              Transfer_Structure => Ftp_Profile.Transfer_Structure,
              Send_Port => Ftp_Profile.Send_Port_Enabled,
              Response => Profile.Get),
end;
```

3. Execute the command.
4. Note the messages that appear in the log file, which may be the current output window.

Transferring Sets of Multiple Files Nonisomorphically

The next example transfers a set of local R1000 text files scattered throughout various libraries. These files are all the files within "!Users.Fred" having the string "Junk" embedded in their filenames. The files will transfer nonisomorphically—that is, without regard to their original relative placements in the local library hierarchy—into a single remote MV library (":Udd:Fred:My_Stuff").

1. Bring up the Ftp.Put_Set command as before. The Command window appears as it did in the previous example.
2. Enter the parameter values as in the previous example, except for the following:

`From_Local_File_Set`	Specify the local files with "!users.fred.?.@junk@".
`Local_Roof`	Set the local roof to "".
`Remote_Roof`	Set the remote roof to ":udd:fred:my_stuff".

After the values are entered, the Command window appears as follows:

```
begin
 Ftp.Put_Set (From_Local_File_Set => "!users.fred.?.@junk@",
              Local_Roof => " ",
              Remote_Roof => ":udd:fred:my_stuff",
              Remote_Machine => "ed",
              Username => "fred",
              Password => "sly_guy",
              Account => Ftp_Profile.Account,
              Remote_Directory => Ftp_Profile.Remote_Directory,
              Remote_Type => Ftp_Profile.Remote_Type,
              Append_To_File => False,
              Transfer_Type => Ftp_Profile.Transfer_Type,
              Transfer_Mode => Ftp_Profile.Transfer_Mode,
              Transfer_Structure => Ftp_Profile.Transfer_Structure,
              Send_Port => Ftp_Profile.Send_Port_Enabled,
              Response => Profile.Get),
end;
```

3. Execute the command and the transfer will proceed as described in previous examples.

FTP Command Summary

A natural order for learning the FTP commands from the reference entries is the following:

Basic Commands	
Ftp.Connect	Forms connection to remote machine for file transfer.
Ftp.Disconnect	Disconnects current connection.
Ftp.Login	Logs user into connected remote machine.
Ftp.Abandon	Forcibly breaks current connection.
Ftp.Store	Copies local file into connected remote machine.
Ftp.Retrieve	Copies remote file from connected remote machine into local machine.
Multiple File Commands	
Ftp.Store_Set	Same as Ftp.Store but operates on file sets and accepts wildcards.
Ftp.Retrieve_Set	Same as Ftp.Retrieve but operates on file sets and accepts wildcards.
Ftp.Retrieve_List	Same as Ftp.Retrieve but accepts list of files.
User Options	
Ftp.Use_Mode	Sets transfer mode (currently only Stream mode supported).
Ftp.Use_Structure	Sets file representation (currently only File supported) for transfer.

Ftp.Use_Type	Sets data representation for transfer (ASCII, Image, Binary). See the Ftp_Defs.Transfer_Type literals for values.
Ftp.Send_Port	Enables verification that local and remote machines are using same data connection.
Data Management	
Ftp.Use_Remote_Type	Sets remote machine type.
Ftp.Use_Remote_Roof	Specifies parent library ("roof") on remote machine containing files to be transferred.
Ftp.Change_Working-_Directory	Sets user's working context on remote machine.
Ftp.Cwd	Sets user's working context on remote machine to be that of Remote_Directory.
Ftp.List	Produces directory listing for remote machine.
Ftp.Delete	Deletes file from remote machine.
Macro Commands	
Ftp.Put	Executes Ftp.Connect, Ftp.Login, Ftp.Store, and Ftp-.Disconnect in sequence.
Ftp.Get	Executes Ftp.Connect, Ftp.Login, Ftp.Retrieve, and Ftp.Disconnect in sequence.
Ftp.Put_Set	Same as Ftp.Put but operates on file sets and accepts wildcards.
Ftp.Get_Set	Same as Ftp.Get but operates on file sets and accepts wildcards.
Ftp.Get_List	Same as Ftp.Get but accepts list of files.
Monitoring Commands	
Ftp.Status	Returns status of connections and commands.
Ftp.Status_All	Displays information for all package FTP connections from local machine.
Ftp_Remote_Status	Returns status from remote machine.
Ftp.Remote_Help	Returns help information from remote server about commands.
Ftp.Show_Profile	Displays FTP parameters in user's switch file.
Ftp.Current_Roof	Returns current value of Ftp.Use_Remote_Roof.
Ftp.Current_Remote-_Type	Returns current value of Ftp.Use_Remote_Type.

Ftp.Use_Type Constants

Ftp.Ascii	Ftp.Ebcdic (not supported)
Ftp.Image	Ftp.Binary
Ftp.Local_Binary	Ftp.Local_Byte
Ftp.Ascii_Cc (not supported)	Ftp.Ebcdic_Cc (not supported)
Ftp.Ascii_Telnet	Ftp.Ebcdic_Telnet (not supported)

Ftp.Use_Mode Constants

Ftp.Stream	Ftp.Record (not supported)
Ftp.Page (not supported)	

Ftp.Use_Structure Constants

Ftp.File

Ftp.Remote_Machine Constants

Ftp.Rational	Ftp.Unix
Ftp.R1000	Ftp.Aos
Ftp.Vms	Ftp.Mvs

package File_Transfer

Package File_Transfer provides a programmatic interface for carrying out file transfers using the File Transfer Protocol (FTP). This package manages the state associated with the connection and carries out the transfer of files and other information. The user acquires resources by opening a connection that is identified by the handle. This handle is used in subsequent commands to identify the connection and its resources.

To use FTP, the user first opens a command connection between the local machine and an FTP server on some remote machine. This connection is used to communicate requests and status between the two machines. Requests are sent across the command connection to the remote server. These requests consist of commands followed by optional arguments. The remote server reads these commands, processes them, and then sends back a response informing the host machine of the outcome of the command. The remote server could have successfully processed the command, unsuccessfully processed it, or not supported the command or some of its options. In general, the communication is in the form of a negotiation in which the local client makes a request and the remote server responds as to whether it has accepted or rejected the request.

Each response sent back contains a three-digit number and some text. The number is used by the local client to determine the outcome of the latest request. The text is intended for the user. The user can use the Read_Response procedure to read these responses.

Both the remote and the local machines maintain information about the various FTP options. When the user wants to change one of these options, a negotiation takes place. If both machines support the option, then the new value takes effect. If one or both of the machines does not support the requested option, the old value remains unchanged.

For the transfer of files, a second connection is used. This connection is formed between the two machines when each file transfer is invoked. The connection is formed by the remote server and requires no special commands from the user. The protocol is that the local client selects a connection and listens on it, waiting for the remote server to connect. A transfer command, such as Ftp.Store, is sent to the remote server, telling it the file to transfer and that a connection is waiting. The remote server connects to the local client and the file transfer occurs. The specification for FTP defines the location of this data connection relative to the command connection so that the remote server knows where the host server will be listening. There is also a command that can be sent to identify this connection. This command allows the local client to clarify the identity of the connection; in some cases, the local client can use a connection other than the specified standard connection.

Several options control the transfer of data between the machines. These options include Mode, which specifies how the bits of data are to be transmitted, and Structure and Type, which define the way in which the file is to be represented. The user can select these options by using the Set_Mode, Set_Structure, and Set_Type commands. There are multiple values for each of these options.

The reading and writing of the local files is carried out by using one of two nonbyte-aligned input/output packages. If the user has set the transfer type to Binary or Local_Binary, the local client assumes that the file contains nonbyte-aligned data and uses package !Io.Polymorphic_Io when manipulating the file. If the user has set the transfer type to Ascii, Ebcdic, Image, or any of the other types, then the file is manipulated using package !Io.Device_Independent_Io.

Each implementation of FTP need not support all FTP options. The local client itself supports only a subset of the options. When a user selects a value for one of these options, the local client first determines whether it can support that option. If the local client does not support the new value, a status code of No_Local_Support is returned and the setting is left at its old value. If the local client does support the requested value, it then sends a command to the remote server requesting that it update its setting for that option to the new value. If the remote server supports the option, it updates its setting and returns a response indicating that it has done so. If it does not support the new value, it returns a response indicating that the value has not been changed. The local client processes the response and, if it was a positive response, updates its local copy of the option to the new value. If a negative response is received, the local copy remains at its old setting. The status code for the command indicates the outcome of each of these transactions.

Once the options have been agreed upon, the file transfer can occur. When the user invokes a transfer command, the local client prepares the second connection over which the file will be transferred. The request is then sent to the remote machine either to receive a file, writing it to the specified location, or to send a copy of the specified file back to the local host. In either case, the remote server initiates the connection for the data connection and the file is transferred.

In addition to the file transfer commands, other FTP commands give the user the ability to query the remote machine for status or help information, get directory listings, or manipulate files on the remote machines. Again, some of these commands are optional for server implementations, and the user may receive a response indicating that the command is not implemented.

procedure Close

```
procedure Close (Connection : Connect_Id);
```

Description

Closes the given connection.

The underlying resources associated with the connection are released and made available for use by other connections.

If the given connection is connected, it is disconnected. If the given connection is already closed or has never been opened, this procedure does nothing.

Parameters

```
Connection :  Connect_Id;
```

Specifies the connection to be closed.

Errors

If the user queries status after executing close, an Ftp_Defs.Status_Code.Invalid-_Use status is returned.

References

procedure Open

function Command_Is_Active

```
function Command_Is_Active (Connection : Connect_Id) return Boolean;
```

Description

Indicates whether a command is currently being processed on the given connection.

A command starts with the sending of the request to the remote machine; it is complete when the completion response is received from the remote machine.

If the connection is not open or is not connected, false is returned.

Parameters

```
Connection : Connect_Id;
```

Specifies the connection.

```
return Boolean;
```

Returns true if a command is currently active on the given connection.

procedure Command_Status

```
procedure Command_Status (Connection :     Connect_Id;
                          Status     : out Ftp_Defs.Status_Code);
```

Description

Returns the final status of the last operation performed.

If the connection is closed, or has never been opened, a status of Ftp_Defs.Status_Code.Invalid_Use is returned. If the connection is no longer connected to a remote machine, or was never connected, then a status of Ftp_Defs.Status_Code.Network_Error is returned. If the command is still in progress, this command blocks until the command completes. Also, all remaining responses are ignored.

Parameters

```
Connection :  Connect_Id;
```

Specifies the connection.

```
Status :  out Ftp_Defs.Status_Code;
```

Returns the outcome of the operation.

Example

```
    declare
        Status : Ftp_Defs.Status_Code;
    begin
        File_Transfer.Send_Username (Connection => Connection,
                                     Username => "Netuser");
        Ftp.Command_Status (Connection => Connection;
                            Status => Status);
        if Status = Ftp_Defs.Need_Password then
           File_Transfer.Send_Password (Connection => Connection,
                                        Password => "Netpass");
           Ftp.Command_Status (Connection => Connection;
                               Status => Status);
           if Status = Ftp_Defs.Successful then
              ...
           else
              ...
           end if;
        end if;
     end;
```

References

package Ftp_Defs

---

procedure Connect

```
procedure Connect (Connection      : Connect_Id;
                   To_Remote_Host  : Transport_Defs.Host_Id;
                   Remote_Socket   : Transport_Defs.Socket_Id  :=
                                             Default_Ftp_Socket);
```

Description

Connects to an FTP server on some remote machine.

The user first must have executed the Open procedure to allocate local resources. If the connection has not been opened, the Connect procedure fails and the Command_Status procedure returns Ftp_Defs.Status_Code.Invalid_Use.

Parameters

`Connection : Connect_Id;`
Specifies the local connection object used to establish the connection.

`To_Remote_Host : Transport_Defs.Host_Id;`
Specifies the host to which the connection is to be formed.

`Remote_Socket : Transport_Defs.Socket_Id := Default_Ftp_Socket;`
Specifies the socket on which the remote host's FTP waits for connections.

Restrictions

The connection must first be opened.

Example

```
    declare
         Status   : Transport_Defs.Status_Code;
         Connection : Connect_Id;
         Remote_Host : constant Transport_Defs.Host_Id := (89,21,1,2);
    begin
         Open (Connection,Status);
         if Status = Transport_Defs.Ok then
            Connect (Connection, Remote_Host);
         end if;
    end;
```

References

procedure Open

TRL, package Transport_Defs

---

type Connect_Id

```
type Connect_Id is private;
```

Description

Specifies the resources in this machine associated with the FTP connection.

References

procedure Close

procedure Connect

procedure Disconnect

procedure Open

function Current_Mode

```
function Current_Mode (Connection : Connect_Id) return Ftp_Defs.Mode_Code;
```

Description

Returns the current transfer mode setting for the given connection.

If the connection is not open, the default mode (Ftp_Defs.Default_Mode) is returned.

Parameters

```
Connection : Connect_Id;
```
Specifies the connection.

```
return Ftp_Defs.Mode_Code;
```
Returns the current mode for transfers on the connection.

References

package Ftp_Defs

function Current_Structure

```
function Current_Structure (Connection : Connect_Id)
                              return Ftp_Defs.Structure_Code;
```

Description

Returns the current transfer structure setting for the given connection.

If the connection is not open or is not connected, the default structure (Ftp_Defs-.Default_Structure) is returned.

Parameters

```
Connection : Connect_Id;
```

Specifies the connection.

```
return Ftp_Defs.Structure_Code;
```

Returns the current structure for transfers on the connection.

References

package Ftp_Defs

function Current_Type

```
function Current_Type (Connection : Connect_Id) return Ftp_Defs.Type_Code;
```

Description

Returns the current transfer type setting for the given connection.

If the connection is not open or is not connected, the default type (Ftp_Defs.Default-_Type) is returned.

Parameters

```
Connection : Connect_Id;
```

Specifies the connection.

```
return Ftp_Defs.Type_Code;
```

Returns the current type for transfers on the connection.

References

package Ftp_Defs

function Data_Host_Id

```
function Data_Host_Id (Connection : Connect_Id)
                                        return Transport_Defs.Host_Id;
```

Description

Returns the local Host_Id for the transport connection that is used to transfer the file data.

The value is the host identifier for the host machine.

Parameters

`Connection : Connect_Id;`
Specifies the connection.

`return Transport_Defs.Host_Id;`
Returns the host identifier.

Restrictions

The connection must be open. If the connection is not open, a null host identifier is returned.

References

procedure Send_Data_Port

TRL, package Transport

TRL, package Transport_Defs

function Data_Socket_Id

```
function Data_Socket_Id (Connection : Connect_Id)
                                    return Transport_Defs.Socket_Id;
```

Description

Returns the local Socket_Id for the transport connection that is used for the transfer of file data.

Parameters

`Connection : Connect_Id;`
Specifies the connection.

`return Transport_Defs.Socket_Id;`
Returns the socket identifier.

Restrictions

The connection must be open. If the connection is not open, a null socket identifier is returned.

References

procedure Send_Data_Port

TRL, package Transport

TRL, package Transport_Defs

constant Default_Ftp_Socket

```
Default_Ftp_Socket  : constant Transport_Defs.Socket_Id  :=
                                             (1 => Ø, 2 => Socket_21);
```

Description

Defines a constant for the Socket_Id for FTP servers.

References

constant Socket_21

function Dio_File_Of

```
function Dio_File_Of (Connection : Connect_Id) return Ftp_Defs.Dio_Pointer;
```

Description

Returns access to the file handle used internally by the FTP transfer service when reading or writing files.

When carrying out nonbinary transfers, the local file transfer server uses a specific file handle when reading from or writing into a local file. These operations are done using package !Io.Device_Independent_Io.

In some cases, a user may want to share access to this file handle. An example could be that the user wants to have a program generate data into a local file and then transfer that file across the network. One way would be to write the data into a named file and then request a file transfer to transfer the file given its name.

Another method would be to open this local file using the file handle from package File_Transfer, write the information into the file, and then reset the file for reading. File transfer is then requested to transfer with no local filename given that informs it that the file handle is already open. If the form expressing a local filename is used, an error occurs because the local server tries to open the named file using the handle that the user already has in use.

Parameters

```
Connection : Connect_Id;
```

Specifies the connection.

```
return Ftp_Defs.Dio_Pointer;
```

Returns an access type pointing to the shared file handle.

Restrictions

Transfer requests that specify a local filename open that file using the server's file handle. If the user already has the file handle in use, the transfer request fails.

Example

```
declare
  The_File : Ftp_Defs.Dio_Pointer;
begin
  The_File := File_Transfer.Dio_File_Of (Connection);
  Device_Independent_Io.Open (File => The_File.All,
                              Mode => Device_Independent_Io.Out_File);
               --- Open local temp file.
     .
     .
     .

  Device_Independent_Io.Write (File => The_File.All,
                               "some data");
               --- Write information to the file.
     .
     .
     .

  Device_Independent_Io.Reset
        (The_File.All,Device_Independent_Io.In_File);
               --- Reset the file so the file transfer server can
               --- read from it.

  File_Transfer.Start_Store (Connection => Connection,
                             Remote_Pathname => "remotefile");
               --- Start the transfer with the form specifying no
               --- local filename.  This form will use the open file
               --- handle.

  File_Transfer.Command_Status (Connection => Connection,
                                Status => Status);
               --- Check on the status of the transfer.
               --- This procedure returns when the transfer
               --- is complete.

  Device_Independent_Io.Close (File => The_File.All);
               --- Since the user had control of the file, it
               --- will need to be closed at the end of the transfer.
end;
```

References

function Pio_File_Of

package Ftp_Defs

procedure Disconnect

```
procedure Disconnect (Connection : Connect_Id);
```

Description

Disconnects the current command connection to the remote host.

If no connection exists, nothing happens.

Parameters

```
Connection :  Connect_Id;
```

Specifies the connection.

Restrictions

If the user queries status after executing disconnect, an Ftp_Defs.Status_Code.Invalid_Use status is returned.

References

procedure Connect

procedure Open

function End_Of_Line

```
function End_Of_Line (Connection : Connect_Id) return Boolean;
```

Description

Returns true when the end of line of a response on the command connection has been reached.

If no command is active and the function is called, then true is returned.

Parameters

```
Connection : Connect_Id;
```

Specifies the connection.

```
return Boolean;
```

Returns the end-of-line indication for responses on the connection.

Restrictions

The connection should be open. If the connection is not open, the End_Of_Line function always returns true.

Example

```
    while not End_of_Response(Connection)  loop
        while not End_of_Line (Connection) loop
            Read_Response(Connection,Message,Count);
            Text_Io.Put (Message(1 .. Count));
        end loop;
        Text_Io.New_Line;
    end loop;
```

References

function End_Of_Response

procedure Read_Response

function End_Of_Response

```
function End_Of_Response (Connection : Connect_Id) return Boolean;
```

Description

Returns true when there are no further responses to the current command.

If no command is active, true is returned.

Parameters

```
Connection : Connect_Id;
```

Specifies the connection.

```
return Boolean;
```

Returns true if no further responses exist.

Restrictions

The connection should be open. If the connection is not open, the End_Of_Response function always returns true.

Example

```
    while not End_of_Response(Connection)  loop
        while not End_Of_Line  (Connection)  loop
            Read_Response(Connection,Message,Count);
            Text_Io.Put  (Message(1  ..  Count));
        end loop;
        Text_Io.New_Line;
    end loop;
```

References

function End_Of_Line

procedure File_Transfer_Status

```
procedure File_Transfer_Status (Connection :     Connect_Id;
                                Status     : out Ftp_Defs.Transfer_Status);
```

Description

Returns the current status of the most recent data transfer request.

If the transfer is active, the status Ftp_Defs.Transfer_Status.In_Progress is returned. If the transfer is complete, the final outcome of the transfer is returned.

Parameters

`Connection :  Connect_Id;`

Specifies the connection.

`Status :  out Ftp_Defs.Transfer_Status;`

Returns the status of the most recent transfer.

procedure Get_Owner

```
procedure Get_Owner (Connection :     Connect_Id;
                     Owner      : out Machine.Job_Id);
```

Description

Returns the owner of the connection.

Parameters

```
Connection :  Connect_Id;
```

Specifies the connection.

```
Owner :  out Machine.Job_Id;
```

Specifies the owner of the connection or, if the connection is not open, Machine.Nil-_Job_Id.

References

procedure Set_Owner

function Is_Connected

```
function Is_Connected (Connection : Connect_Id) return Boolean;
```

Description

Determines whether the specified connection is connected to a remote machine.

Parameters

```
Connection : Connect_Id;
```
Specifies the connection.

```
return Boolean;
```
Returns true if the specified connection is linked to a remote machine.

Restrictions

If the connection is closed or has never been opened, false is returned.

References

procedure Connect

procedure Disconnect

function Is_Logged_In

```
function Is_Logged_In (Connection : Connect_Id) return Boolean;
```

Description

Returns, for the given connection, that the user has been successfully logged into the remote host.

Parameters

```
Connection : Connect_Id;
```

Specifies the connection.

```
return Boolean;
```

Returns true if the connection is successfully logged in.

function Is_Open

```
function Is_Open (Connection : Connect_Id) return Boolean;
```

Description

Checks whether the specified connection is currently open.

If the connection is closed or has never been opened, false is returned.

Parameters

```
Connection : Connect_Id;
```
Specifies the connection.

```
return Boolean;
```
Returns true if the specified connection is open.

References

procedure Close

procedure Open

function Last_Transfer_Length

```
function Last_Transfer_Length (Connection : Connect_Id) return Natural;
```

Description

Returns a count of the number of bytes sent or received during the last file transfer.

Parameters

```
Connection :  Connect_Id;
```

Specifies a connection.

```
return Natural;
```

Returns the number of bytes that were transferred during the last file transfer.

Restrictions

The connection should be open. If the connection is not open, no error occurs but a length of 0 is returned.

function Last_Transfer_Time

```
function Last_Transfer_Time (Connection : Connect_Id) return Duration;
```

Description

Returns the time taken to complete the last file transfer.

Parameters

```
Connection : Connect_Id;
```

Specifies a connection.

```
return Duration;
```

Returns the time, in seconds, of the last transfer.

Restrictions

The connection should be open. If the connection is not open or if no transfer has taken place, a value of 1.0 seconds is returned.

function Most_Recent_Command

```
function Most_Recent_Command (Connection : Connect_Id)
                                         return Ftp_Defs.Ftp_Commands;
```

Description

Indicates which command was last executed.

This can be an active command or the last-finished command.

Parameters

`Connection : Connect_Id;`
Specifies a connection.

`return Ftp_Defs.Ftp_Commands;`
Returns a value indicating the most recent command.

References

procedure Command_Status

function Most_Recent_Command_Status

```
function Most_Recent_Command_Status (Connection : Connect_Id)
                                          return Ftp_Defs.Status_Code;
```

Description

Returns the status value for the command most recently processed.

If the command is currently active, the status Ftp_Defs.Status_Code.Command-_In_Progress is returned.

Parameters

`Connection : Connect_Id;`
Specifies the connection.

`return Ftp_Defs.Status_Code;`
Returns the status for the most recent command.

Restrictions

The connection should be open. If not, the status is Ftp_Defs.Status_Code.Invalid-_Use.

References

procedure Command_Status

function Most_Recent_Response_Code

```
function Most_Recent_Response_Code (Connection : Connect_Id)
                                                    return Natural;
```

Description

Returns the response code from the most recent response.

FTP responses from the remote host have a three-digit code number at the beginning intended for use by a program to determine the next processing step. This function makes those values visible to the user.

Parameters

`Connection : Connect_Id;`

Specifies the connection.

`return Natural;`

Returns the value of the most recent response code.

Restrictions

If the connection is not open, the default value of 000 is returned.

function Most_Recent_Transfer_Status

```
function Most_Recent_Transfer_Status (Connection : Connect_Id)
                                  return Ftp_Defs.Transfer_Status;
```

Description

Returns the current status of the most recent file transfer.

If the transfer is still active, the value Ftp_Defs.Transfer_Status.In_Progress is returned.

Parameters

`Connection : Connect_Id;`

Specifies the connection.

`return Ftp_Defs.Transfer_Status;`

Returns the status for the most recent command.

Restrictions

If the connection is not open, or until a transfer has occurred, the value returned is Ftp_Defs.Transfer_Status.Unknown_Error.

References

package Ftp_Defs

constant Null_Connect_Id

```
Null_Connect_Id : constant Connect_Id;
```

Description

Defines a constant representing the value for the socket number on which the FTP server waits for a command connection.

procedure Open

```
procedure Open (Connection : out Connect_Id;
                Status     : out Transport_Defs.Status_Code);
```

Description

Allocates the resources required to form a connection.

Once opened, the connection can be connected and disconnected many times. It continues to be associated with the same local resources.

Parameters

`Connection : out Connect_Id;`

Returns a Connect_Id that can be used to denote the connection in subsequent calls to other procedures.

`Status : out Transport_Defs.Status_Code;`

Returns the outcome of the operation.

References

procedure Close

procedure Connect

function Pasv_Data_Host

```
function Pasv_Data_Host (Connection : Connect_Id)
                                       return Transport_Defs.Host_Id;
```

Description

Returns the Host_Id associated with the data connection the remote user is waiting on after the Send_Pasv procedure has successfully executed.

This function is used for third-party transfers (transfer of a file between two remote machines, directed by the user's local machine).

Parameters

`Connection : Connect_Id;`
Specifies the connection.

`return Transport_Defs.Host_Id;`
Returns the host identity of the remote data link.

Restrictions

The Send_Pasv command requests the remote server to listen on its data connection rather than initiate the connect when a data transfer is started. The remote machine returns the identity of the connection in the response to the Send_Pasv command. When interpreting this response, the local process may be unable to find the host identifier. In this case, Transport_Defs.Null_Host_Id is returned. A command status of Ftp_Defs.Status_Code.Local_Pasv_Error indicates that the value could not be parsed from the response.

References

procedure Send_Pasv

TRL, package Transport_Defs

function Pasv_Data_Socket

```
function Pasv_Data_Socket (Connection : Connect_Id)
                                 return Transport_Defs.Socket_Id;
```

Description

Returns the Socket_Id associated with the data link being listened to by the remote machine after the Send_Pasv procedure has successfully executed.

Parameters

`Connection : Connect_Id;`

Specifies a connection.

`return Transport_Defs.Socket_Id;`

Returns the socket identity of the remote data link.

Restrictions

The Send_Pasv command requests the remote server to listen on its data link rather than initiate the connect when a data transfer is started. The remote machine returns the identity of the data link in the response to the Send_Pasv command. When interpreting this response, the local process may be unable to interpret this response to find the Socket_Id. In this case, Transport_Defs.Null_Socket_Id is returned. A command status of Ftp_Defs.Status_Code.Local_Pasv_Error indicates that the value could not be parsed from the response.

References

procedure Send_Pasv

TRL, package Transport_Defs

function Pio_File_Of

```
function Pio_File_Of (Connection : Connect_Id) return Ftp_Defs.Pio_Pointer;
```

Description

Returns access to the file handle used internally by the FTP transfer service when reading or writing binary files.

This function gives the user visibility to this handle so that a user program can open and manipulate the file used by the transfer service.

When transferring files, the transfer service uses a specific file handle for access to the file being transferred. This function allows a user program to share access to this handle. In this way, a user can retrieve data into a local file and still have control over the file.

It is important that the user not manipulate the file handle while the file transfer is in progress. Also, once the transfer has completed, the user must close the file handle. If the user has this handle still open, the server is unable to open the new file.

Parameters

`Connection : Connect_Id;`
Specifies the connection.

`return Ftp_Defs.Pio_Pointer;`
Returns an access type pointing to the shared file handle.

Restrictions

Binary transfer requests that specify a local filename open that file using the server's file handle. If the user already has the file handle in use, the transfer request fails.

Example

At this point, there is a connection set up for binary transfers:

```
declare
  The_File : Ftp_Defs.Pio_Pointer;
begin
  The_File := File_Transfer.Pio_File_Of (Connection);

  Polymorphic_Io.Open (The_Handle => The_File.Handle,
                       Mode => Polymorphic_Io.Read_Only,
                       File_Name => "Movefile",
                       Status => Poly_Status);
  The_File.Position := Polymorphic_Io.First;

  File_Transfer.Start_Store (Connection => Connection,
                             Remote_Pathname => "Remotefile");
  File_Transfer.Command_Status (Connection => Connection,
                                Status => Status);
  Polymorphic_Io.Close (File => The_File.Handle,
                        Status => Poly_Status);
end;
```

References

function Dio_File_Of

package Ftp_Defs

procedure Read_Response

```
procedure Read_Response (Connection :     Connect_Id;
                         Message    : out String;
                         Count      : out Natural);
```

Description

Returns the text messages received from the FTP server on the remote host.

The messages still contain the code values found at the start of the messages. The messages are line-oriented.

Parameters

`Connection : Connect_Id;`

Specifies the connection.

`Message : out String;`

Specifies the object that receives the value read.

`Count : out Natural;`

Specifies the length of the value read.

Restrictions

If the connection is not open, the Read_Response procedure returns a null message with a count of 0.

Example

```
    while not End_Of_Response(Connection)  loop
        while not End_Of_Line (Connection) loop
            Read_Response(Connection,Message,Count);
            Text_Io.Put (Message(1 .. Count));
        end loop;
        Text_Io.New_Line;
    end loop;
```

References

function End_Of_Line

function End_Of_Response

function Remote_Host_Id

```
function Remote_Host_Id (Connection : Connect_Id)
                                        return Transport_Defs.Host_Id;
```

Description

Returns the Host_Id for the remote server to which the connection is currently connected.

Parameters

```
Connection :  Connect_Id;
```

Specifies the connection.

```
return Transport_Defs.Host_Id;
```

Specifies the host identifier of the remote machine.

Restrictions

If the connection is not open or not connected, the value Transport_Defs.Null_Host-_Id is returned.

References

procedure Connect

TRL, package Transport_Defs

procedure Send_Account

```
procedure Send_Account (Connection : Connect_Id;
                        Account    : String);
```

Description

Transmits account information to the remote host that may be required for login or file access.

Parameters

```
Connection :  Connect_Id;
```

Specifies a connection.

```
Account :  String;
```

Specifies the value for the account information.

Restrictions

The connection must first be open and connected to a remote host. If the connection is not open, the Command_Status procedure returns Ftp_Defs.Status_Code.Invalid_Use. If the connection is not connected, the Command_Status procedure returns Ftp_Defs.Status_Code.Network_Error.

procedure Send_Cwd

```
procedure Send_Cwd (Connection      : Connect_Id;
                    Remote_Pathname : String);
```

Description

Sends a request for the remote server to change the default directory to that specified by the Remote_Pathname parameter.

Parameters

```
Connection :  Connect_Id;
```

Specifies a connection.

```
Remote_Pathname :  String;
```

Specifies the value for the remote context to which the user wants the directory to be set.

Restrictions

The remote pathname must be a legal directory pathname for the remote host. The connection must first be open and connected to a remote host. If the connection is not open, the Command_Status procedure returns Ftp_Defs.Status_Code.Invalid_Use. If the connection is not connected, the Command_Status procedure returns Ftp_Defs.Status_Code.Network_Error.

procedure Send_Data_Port

```
procedure Send_Data_Port (Connection : Connect_Id;
                          Host       : Transport_Defs.Host_Id;
                          Socket     : Transport_Defs.Socket_Id);
```

Description

Transmits the identity of the local data link to the remote server.

In normal use, the user supplies the host and socket information through the Data_Host_Id and Data_Socket_Id functions. There are cases, however, in which a user may be using the local server as an intermediary for communications between two remote servers. In this case, the data port sent is for one of the remote machines.

Parameters

`Connection : Connect_Id;`

Specifies a connection.

`Host : Transport_Defs.Host_Id;`

Specifies the Internet host address.

`Socket : Transport_Defs.Socket_Id;`

Specifies the TCP socket (or port) address.

Restrictions

The connection must first be open and connected to a remote host. If the connection is not open, the Command_Status procedure returns Ftp_Defs.Status_Code.Invalid_Use. If the connection is not connected, the Command_Status procedure returns Ftp_Defs.Status_Code.Network_Error. If the host and socket values supplied do not specify a data connection that is waiting for a connection during file transfers, errors occur. In normal use, the user should follow the example below.

Example

This example, which causes a new socket to be selected for the transfer, is recommended for multiple transfers:

```
    Send_Data_Port (Connection => Connection,
                    Host => Data_Host_Id (Connection),
                    Socket => Data_Socket_Id (Connection);


    Send_Data_Port (Connection => Connection,
                    Host => Transport_Defs.Null_Host_Id,
                    Socket => Transport_Defs.Null_Host_Id;
```

References

procedure Send_Pasv

package Transport_Defs

procedure Send_Delete

```
procedure Send_Delete (Connection      : Connect_Id;
                       Remote_Filename : String);
```

Description

Sends a request for the remote server to delete the specified file.

Parameters

`Connection : Connect_Id;`

Specifies a connection.

`Remote_Filename : String;`

Specifies the filename of the file to be deleted.

Restrictions

The connection must first be open and connected to a remote host. If the connection is not open, the Command_Status procedure returns Ftp_Defs.Status_Code.Invalid_Use. If the connection is not connected, the Command_Status procedure returns Ftp_Defs.Status_Code.Network_Error.

procedure Send_Help_Request

```
procedure Send_Help_Request (Connection : Connect_Id;
                             Help_On    : String     := "");
```

Description

Requests the server to send helpful information regarding its implementation status over the command connection.

The response to the command is read using the Read_Response procedure.

Parameters

`Connection : Connect_Id;`

Specifies a connection.

`Help_On : String := "";`

Specifies a value for which specific help is requested.

Restrictions

The connection must first be open and connected to a remote host. If the connection is not open, the Command_Status procedure returns Ftp_Defs.Status_Code.Invalid_Use. If the connection is not connected, the Command_Status procedure returns Ftp_Defs.Status_Code.Network_Error.

References

procedure Read_Response

procedure Send_Password

```
procedure Send_Password (Connection : Connect_Id;
                         Password   : String);
```

Description

Sends the password associated with the username previously sent.

This procedure must immediately follow the Send_Username procedure when sent. A command status of Ftp_Defs.Status_Code.Need_Password following Send_Username indicates that the password is needed.

Parameters

```
Connection :  Connect_Id;
```

Specifies a connection.

```
Password :  String;
```

Specifies the password for the user.

Restrictions

This procedure must immediately follow the Send_Username procedure. The connection must first be open and connected to a remote host. If the connection is not open, the Command_Status procedure returns Ftp_Defs.Status_Code.Invalid_Use. If the connection is not connected, the Command_Status procedure returns Ftp_Defs.Status_Code.Network_Error.

Example

```
declare
  Status : Ftp_Defs.Status_Code;
begin
  Send_Username (Connection,"Foo");
  Command_Status (Connection,Status);
  if Status = Ftp_Defs.Need_Password then
     Send_Password (Connection, "Bar");
  end if;
end;
```

References

procedure Send_Username

procedure Send_Pasv

```
procedure Send_Pasv (Connection : Connect_Id);
```

Description

Requests the remote server to go into passive mode on its data port.

This procedure is used in third-party transfers to put one of the servers into the passive mode. The returned port identity is sent to the partner server to complete the connection using the Send_Data_Port procedure call. The remote returns the identity of this port and the local server attempts to read this information. If successful, the Pasv_Data_Host and Pasv_Data_Socket functions return the Internet address for the remote data port. A command status of Local_Pasv_Error indicates that the remote server acted on the request but encountered problems trying to interpret the response.

For a discussion of third-party transfers, see the ARPA RFC-765 documentation for the File Transfer Protocol Specification.

Parameters

`Connection : Connect_Id;`

Specifies a connection.

Restrictions

The connection must first be open and connected to a remote host. If the connection is not open, the Command_Status procedure returns Ftp_Defs.Status_Code.Invalid_Use. If the connection is not connected, the Command_Status procedure returns Ftp_Defs.Status_Code.Network_Error.

References

function Pasv_Data_Host

function Pasv_Data_Socket

procedure Send_Data_Port

procedure Send_Quit

```
procedure Send_Quit (Connection : Connect_Id);
```

Description

Sends a request to log out the session from the remote server.

If the remote server accepts and processes this command, it disconnects the command connection from its side after sending the response. After reading the responses, the user should use the Is_Connected function to determine that the line was successfully disconnected.

Parameters

```
Connection : Connect_Id;
```

Specifies a connection.

References

function Is_Connected

procedure Send_Site_Command

```
procedure Send_Site_Command (Connection : Connect_Id;
                             Argument   : String);
```

Description

Sends a site-specific request to the remote server.

Extensions and site-specific operations are performed using the FTP protocol "SITE" command.

Parameters

```
Connection : Connect_Id;
```

Specifies a connection.

```
Argument : String;
```

Specifies the argument for the FTP "SITE" command.

Restrictions

The connection must first be open and connected to a remote host. If the connection is not open, the Command_Status procedure returns Ftp_Defs.Status_Code.Invalid_Use. If the connection is not connected, the Command_Status procedure returns Ftp_Defs.Status_Code.Network_Error.

The user should be careful that the FTP protocol "SITE" command does not have any side effects, such as starting a transfer, because the local server will be unprepared for it.

procedure Send_Status_Request

```
procedure Send_Status_Request (Connection : Connect_Id;
                               Status_On  : String     := "");
```

Description

Sends a request for the status of the remote server process.

Parameters

`Connection : Connect_Id;`

Specifies a connection.

`Status_On : String := "";`

Specifies an additional argument to the status request. If null, general status is returned. If nonnull, file/directory information is returned for the given argument.

Restrictions

The connection must first be open and connected to a remote host. If the connection is not open, the Command_Status procedure returns Ftp_Defs.Status_Code.Invalid_Use. If the connection is not connected, the Command_Status procedure returns Ftp_Defs.Status_Code.Network_Error.

procedure Send_Username

```
procedure Send_Username (Connection : Connect_Id;
                         Username   : String);
```

Description

Sends username information to the remote FTP server.

This is the first command to be sent after a successful connection has been made.

Parameters

`Connection : Connect_Id;`
Specifies a connection.

`Username : String;`
Specifies the username to be used when logging into the current session.

Restrictions

The connection must first be open and connected to a remote host. If the connection is not open, the Command_Status procedure returns Ftp_Defs.Status_Code.Invalid_Use. If the connection is not connected, the Command_Status procedure returns Ftp_Defs.Status_Code.Network_Error.

References

procedure Connect

procedure Send_Password

procedure Send_Verbatim

```
procedure Send_Verbatim (Connection : Connect_Id;
                         Command    : String);
```

Description

Sends an arbitrary FTP protocol command.

The Command parameter is sent across the command connection unchanged. This command must be a simple command that has no side effects, such as causing the server to try to transfer a file. The local client does not act on the responses but merely passes them to the user.

Parameters

`Connection :  Connect_Id;`
Specifies a connection.

`Command :  String;`
Specifies the value to be sent to the remote server as a command.

Restrictions

This procedure should not cause any side effects, such as a file transfer. The local client does not act on the responses.

The connection must first be open and connected to a remote host. If the connection is not open, the Command_Status procedure returns Ftp_Defs.Status_Code.Invalid_Use. If the connection is not connected, the Command_Status procedure returns Ftp_Defs.Status_Code.Network_Error.

procedure Set_Allocation

```
procedure Set_Allocation (Connection : Connect_Id;
                          Pages      : Natural;
                          Records    : Natural);
```

Description

Reserves storage space for the next file to be stored.

This command may be required by some servers to reserve sufficient storage to accommodate the new file to be transferred. The argument specifies the number of bytes to be reserved. This command is not needed for Rational machines.

Parameters

`Connection : Connect_Id;`

Specifies a connection.

`Pages : Natural;`

Specifies the size in bytes to be reserved for the file.

`Records : Natural;`

Specifies the size in bytes of a page or record (optional).

Restrictions

The connection must first be open and connected to a remote host. If the connection is not open, the Command_Status procedure returns Ftp_Defs.Status_Code.Invalid_Use. If the connection is not connected, the Command_Status procedure returns Ftp_Defs.Status_Code.Network_Error.

procedure Set_Mode

```
procedure Set_Mode (Connection : Connect_Id;
                    New_Mode   : Ftp_Defs.Mode_Code);
```

Description

Sends a request for the transfer mode setting to be changed to a specified value.

The local server first checks to see that it supports the mode; if it does, the request is sent to the remote server. If a positive response is received, the local setting is updated. If either server does not support the new option, the current value is left unchanged.

A command status of Ftp_Defs.Status_Code.No_Local_Support indicates that the local server does not support the option. A status of Command_Not_Implemented or Param_Not_Implemented indicates that the remote server does not support the option.

Parameters

`Connection : Connect_Id;`

Specifies a connection.

`New_Mode : Ftp_Defs.Mode_Code;`

Specifies the new transfer mode.

Restrictions

The connection must first be open and connected to a remote host. If the connection is not open, the Command_Status procedure returns Ftp_Defs.Status_Code.Invalid_Use. If the connection is not connected, the Command_Status procedure returns Ftp_Defs.Status_Code.Network_Error.

References

package Ftp_Defs

procedure Set_Owner

```
procedure Set_Owner (Connection : in Connect_Id;
                     Owner      :    Machine.Job_Id);
```

Description

Sets the owner of a connection.

Each connection is owned by some job. When a job terminates, all connections owned by it are closed automatically.

By default, a connection is owned by the job that owned it.

Parameters

`Connection :  in Connect_Id;`
Specifies a connection.

`Owner :  Machine.Job_Id;`
Specifies the new owner of the connection.

Restrictions

The connection must first be open.

References

procedure Get_Owner

procedure Set_Structure

```
procedure Set_Structure (Connection    : Connect_Id;
                         New_Structure : Ftp_Defs.Structure_Code);
```

Description

Sends a request for the transfer structure setting to be changed to a specified value.

The local server first checks to see that it supports the mode; if it does, the request is sent to the remote server. If a positive response is received, the local setting is updated. If either server does not support the new option, the current value is left unchanged.

A command status of Ftp_Defs.Status_Code.No_Local_Support indicates that the local server does not support the option. A status of Command_Not_Implemented or Param_Not_Implemented indicates that the remote server does not support the option.

Parameters

`Connection : Connect_Id;`

Specifies a connection.

`New_Structure : Ftp_Defs.Structure_Code;`

Specifies the new transfer structure.

Restrictions

The connection must first be open and connected to a remote host. If the connection is not open, the Command_Status procedure returns Ftp_Defs.Status_Code.Invalid_Use. If the connection is not connected, the Command_Status procedure returns Ftp_Defs.Status_Code.Network_Error.

References

package Ftp_Defs

procedure Set_Type

```
procedure Set_Type (Connection : Connect_Id;
                    New_Type   : Ftp_Defs.Type_Code);
```

Description

Sends a request for the transfer type setting to be changed to a specified value.

The local server first checks to see that it supports the mode; if it does, the request is sent to the remote server. If a positive response is received, the local setting is updated. If either server does not support the new option, the current value is left unchanged.

A command status of Ftp_Defs.Status_Code.No_Local_Support indicates that the local server does not support the option. A status of Command_Not_Implemented or Param_Not_Implemented indicates that the remote server does not support the option.

Parameters

`Connection : Connect_Id;`

Specifies a connection.

`New_Type : Ftp_Defs.Type_Code;`

Specifies the new transfer type.

Restrictions

The connection must first be open and connected to a remote host. If the connection is not open, the Command_Status procedure returns Ftp_Defs.Status_Code.Invalid_Use. If the connection is not connected, the Command_Status procedure returns Ftp_Defs.Status_Code.Network_Error.

References

package Ftp_Defs

constant Socket_21

```
Socket_21 : constant := 21;
```

Description

Defines a constant representing the value for the socket number on which the FTP server waits for a command connection.

procedure Start_Directory_List

```
procedure Start_Directory_List (Connection      : Connect_Id;
                                Remote_Pathname : String;
                                Verbose         : Boolean    := False);

procedure Start_Directory_List (Connection      : Connect_Id;
                                Local_Filename  : String;
                                Remote_Pathname : String;
                                Verbose         : Boolean    := False);
```

Description

Sends a request for the remote server to transfer a copy of the directory listing associated with the Remote_Pathname parameter as a file transfer.

The first form assumes that the user has already opened the file handle with a call to !Io.Device_Independent_Io.Open, using Dio_File_Of(Connection).All to denote the file handle.

The second form opens the file and writes the contents of the transfer into the file named by the Local_Filename parameter.

In both cases, the local server copies the data transferred into the local file.

Parameters

`Connection : Connect_Id;`

Specifies a connection.

`Local_Filename : String;`

Specifies the filename for the local copy of the directory listing.

`Remote_Pathname : String;`

Specifies the remote pathname for which a directory listing is to be made.

`Verbose : Boolean := False;`

Returns, if true, a verbose listing containing directory information such as size. When false, it returns a list containing only the filenames, one per line.

Restrictions

This procedure should be used only with transfer types of Ascii or Ebcdic. The remote pathname should use a legal pathname or template characters for the remote machine.

References

function Dio_File_Of

procedure Start_Retrieve

```
procedure Start_Retrieve (Connection       : Connect_Id;
                          Remote_Filename  : String);

procedure Start_Retrieve (Connection       : Connect_Id;
                          Local_Filename   : String;
                          Remote_Filename  : String;
                          Append           : Boolean     := False);
```

Description

Sends a request for the remote server to transfer a copy of the file specified by the Remote_Filename parameter to the local server.

The first form assumes that the user has opened the file handle for output using the appropriate open (Binary and Local_Binary transfers use package !Io.Polymorphic_Io; the remainder use package !Io.Device_Independent_Io).

In the second form, the local server stores the data in the file specified by Local_Filename.

In both cases, the local server copies the data transferred into the local file.

For multiple retrieves, the user should use the Send_Data_Port command with the parameter values Null_Host_Id and Null_Socket_Id. This will assign new sockets for the transfer.

Parameters

`Connection : Connect_Id;`

Specifies a connection.

`Local_Filename : String;`

Specifies the destination of a file copied from the remote server.

`Remote_Filename : String;`

Specifies the remote file to be copied.

`Append : Boolean := False;`

Specifies, if true, that the file copied be appended to the local file.

Restrictions

The connection must first be open and connected to a remote host. If the connection is not open, the Command_Status procedure returns Ftp_Defs.Status_Code.Invalid_Use. If the connection is not connected, the Command_Status procedure returns Ftp_Defs.Status_Code.Network_Error. Only one transfer can be active at any given time.

References

function Dio_File_Of

function Pio_File_Of

procedure Send_Data_Port

---

procedure Start_Store

```
procedure Start_Store (Connection      : Connect_Id;
                       Remote_Filename : String;
                       Append          : Boolean     := False);

procedure Start_Store (Connection      : Connect_Id;
                       Local_Filename  : String;
                       Remote_Filename : String;
                       Append          : Boolean     := False);
```

Description

Sends a request for the remote server to accept a copy of the file specified by the Remote_Filename parameter transferred from the local server.

The first form assumes that the user has opened the file handle for input using the appropriate open (Binary and Local_Binary transfers use package !Io.Polymorphic-_Io; the remainder use package !Io.Device_Independent_Io).

In the second form, the local server reads the contents of the file specified by the Local_Filename parameter for transfer to the remote server.

For multiple stores, the user should use the Send_Data_Port command with the parameter values Null_Host_Id and Null_Socket_Id. This will assign new sockets for the transfer.

Parameters

`Connection : Connect_Id;`

Specifies a connection.

`Local_Filename : String;`

Specifies the file to be copied to the remote server.

`Remote_Filename : String;`

Specifies the filename at the remote server for the destination of the file.

`Append : Boolean := False;`

Specifies, if true, that the contents of the file sent to the remote server should be appended to the remote file if it already exists.

Restrictions

The connection must first be open and connected to a remote host. If the connection is not open, the Command_Status procedure returns Ftp_Defs.Status_Code.Invalid_Use. If the connection is not connected, the Command_Status procedure returns Ftp_Defs.Status_Code.Network_Error. Only one transfer can be active at any given time.

References

function Dio_File_Of

function Pio_File_Of

procedure Send_Data_Port

function Transfer_Is_Active

```
function Transfer_Is_Active (Connection : Connect_Id) return Boolean;
```

Description

Returns true if a file transfer is currently active.

Parameters

```
Connection : Connect_Id;
```
Specifies a connection.

```
return Boolean;
```
Returns true if a transfer is currently in progress.

end File_Transfer;

RATIONAL

package Ftp

Package Ftp provides commands for carrying out file transfer to and from remote machines, using the U.S. Department of Defense File Transfer Protocol (FTP).

FTP is an interactive, session-oriented transfer protocol. The user carries out a file transfer by first establishing a connection to the remote machine over which requests are sent and responses returned. Once this connection has been formed, the remote server usually requires that the user supply username and password information, much as if the user were logging onto the machine itself. Some machines may require additional information associating the user with a given account on the remote machine.

When the user has gained access to the remote machine, file transfers can then take place. These transfers occur over a second connection formed between the two machines when each file transfer is invoked.

Several options control the transfer of data between the machines. These options include Mode, which specifies how the bits of the data are to be transmitted, and Structure and Type, which are used to define the way in which the file is to be represented. The user can select these options as parameters to various commands or by using the specific Use_Mode, Use_Structure, and Use_Type commands. There are multiple values for each of these options. Each implementation of FTP need not support all of the choices. The R1000 implementation, for example, supports only a subset of the options. When a user selects a value for one of these options, the client software first determines whether it can support that option. If the client software does not support the new value, the user is informed and the setting is left at its old value. If the local client software does support the requested value, it sends a command to the remote server requesting that it update its setting for that option to the new value. If the remote server supports the option, it updates its setting and returns a response indicating that it has done so. If it does not support the new value, it returns a response indicating that the value has not been changed. The local client processes the response and, if it was a positive response, updates its local copy of the option to the new value. If a negative response is received, the local copy remains at its old setting. The user receives log information indicating the outcome of each of these transactions.

Once the options have been agreed upon, the file transfer can occur. When the user invokes a transfer command, the server prepares the second connection, over which the file will be transferred. The request is then sent to the remote machine either to receive a file, writing it into the specified location, or to send a copy of the specified file back to the local host. In either case, the remote server connects the data connection and the file is transferred.

In addition to the file transfer commands, other commands in package Ftp query the remote machine for status or help information, get directory listings, or manipulate files on the remote machine. Again, some of these commands are optional for server implementations, and the user may receive a response indicating that the command is not implemented.

Some commands (Get, Get_List, Get_Set, Put, Put_Set) bundle several operations into one command. These commands go through the complete cycle of connecting to the remote machine, logging in, transferring files, and finally disconnecting from the remote server. These commands allow the user to carry out all the steps for a file transfer in one command.

Some of the transfer commands can move many files at a time (Get_List, Get_Set, Put_Set, Retrieve_List, Retrieve_Set, and Store_Set). The _Set commands accept a source filename with wildcards, resolve that name on the source machine, and move all the files to which the wildcard expands. The _List commands read a list of remote filenames from a local file and transfer all of the named files.

procedure Abandon

```
procedure Abandon (Response : Profile.Response_Profile := Profile.Get);
```

Description

Causes the connection to the remote host to be broken and local resources freed.

This procedure is used when the remote server has become unresponsive.

Parameters

```
Response :  Profile.Response_Profile  := Profile.Get;
```

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

References

procedure Disconnect

constant Aos

```
Aos : constant Ftp_Name_Map.Machine_Type  := Ftp_Name_Map.Aos;
```

Description

Defines a constant for a Unix machine that imports the Machine_Type value of Aos from Ftp_Name_Map.

This constant is used to indicate that the transfer is to/from a Data General computer using the AOS operating system.

constant Ascii

```
Ascii : constant Ftp_Defs.Type_Code := Ftp_Defs.Ascii;
```

Description

Defines a constant that imports the Type_Code value of Ascii from Ftp_Defs.

This constant is used to indicate that the file being transferred is an ASCII text file.

constant Ascii_Cc

```
Ascii_Cc : constant Ftp_Defs.Type_Code := Ftp_Defs.Ascii_Cc;
```

Description

Defines a constant that imports the Type_Code value of Ascii_Cc from Ftp_Defs.

This constant is used to indicate that the file being transferred is an ASCII text file with ASA (FORTRAN) vertical format control.

constant Ascii_Telnet

```
Ascii_Telnet : constant Ftp_Defs.Type_Code := Ftp_Defs.Ascii_Telnet;
```

Description

Defines a constant that imports the Type_Code value of Ascii_Telnet from Ftp-_Defs.

This constant is used to indicate that the file being transferred is an ASCII text file with Telnet vertical format control.

constant Binary

```
Binary : constant Ftp_Defs.Type_Code := Ftp_Defs.Binary;
```

Description

Defines a constant that imports the Type_Code value of Binary from Ftp_Defs.

This constant is used to indicate that the file being transferred is binary data that are not byte-aligned.

constant Block

```
Block : constant Ftp_Defs.Mode_Code := Ftp_Defs.Block;
```

Description

Defines a constant that imports the Mode_Code value of Block from Ftp_Defs.

This constant is used to indicate that the file is transmitted as a series of data blocks preceded by one or more header bytes.

procedure Change_Working_Directory

```
procedure Change_Working_Directory
(Remote_Directory : String                   :=
                                              Ftp_Profile.Remote_Directory;
 Response         : Profile.Response_Profile := Profile.Get;
 Account          : String                   := Ftp_Profile.Account);
```

Description

Sets the user's working context on the remote machine.

Parameters

`Remote_Directory : String := Ftp_Profile.Remote_Directory;`

Specifies the new directory context.

`Response : Profile.Response_Profile := Profile.Get;`

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

`Account : String := Ftp_Profile.Account;`

Specifies the account information supplied to the remote machine when it requires account information. The default value is obtained from the Account switch in the user's session switch file.

Restrictions

The user must first have connected and logged into a remote server before issuing this command. If the session does not have an active FTP connection, an error message is produced and the command has no effect.

References

procedure Cwd

constant Compressed

```
Compressed : constant Ftp_Defs.Mode_Code := Ftp_Defs.Compressed;
```

Description

Defines a constant that imports the Mode_Code value of Compressed from Ftp-_Defs.

This constant is used to indicate that the file is compressed. It sends replication of bytes and filler as control information describing the number of repetitions of a given byte or filler.

procedure Connect

```
procedure Connect
(To_Machine          : String                      :=
                                                   Ftp_Profile.Remote_Machine;
 Auto_Login          : Boolean                     := Ftp_Profile.Auto_Login;
 Username            : String                      := Ftp_Profile.Username;
 Password            : String                      := Ftp_Profile.Password;
 Account             : String                      := Ftp_Profile.Account;
 Remote_Directory    : String                      :=
                                                 Ftp_Profile.Remote_Directory;
 Remote_Roof         : String                      := Ftp_Profile.Remote_Roof;
 Remote_Type         : Ftp_Name_Map.Machine_Type   := Ftp_Profile.Remote_Type;
 Transfer_Type       : Ftp_Defs.Type.Code          :=
                                                   Ftp_Profile.Transfer_Type;
 Transfer_Mode       : Ftp_Defs.Mode_Code          :=
                                                   Ftp_Profile.Transfer_Mode;
 Transfer_Structure  : Ftp_Defs.Structure_Code     :=
                                              Ftp_Profile.Transfer_Structure;
 Send_Port           : Boolean                     :=
                                               Ftp_Profile.Send_Port_Enabled;
 Response            : Profile.Response_Profile    := Profile.Get);
```

Description

Forms a connection to a remote machine for an FTP file transfer session.

If the user has specified automatic login, the username, password, and account information are used to log onto the remote server. If the user specifies that automatic login is not to occur, only the connection is formed, and the user must supply the login information later.

Parameters

`To_Machine : String := Ftp_Profile.Remote_Machine;`

Specifies the machine to which to make the connection. The default is to use the value specified in the user's switch file.

`Auto_Login : Boolean := Ftp_Profile.Auto_Login;`

Specifies automatic login. If true, the server attempts to log in the user automatically using the username, password, and account information supplied. The default is to use the value specified in the Auto_Login switch in the session switch file.

```
Username : String := Ftp_Profile.Username;
```

Specifies the username to be used when performing Auto_Login to a remote machine. If Auto_Login is false, this parameter is ignored.

```
Password : String := Ftp_Profile.Password;
```

Specifies the password to be used if needed for login to the remote FTP server. If Auto_Login is false, this parameter is ignored.

```
Account : String := Ftp_Profile.Account;
```

Specifies the account information supplied to the remote machine when it requires account information. If Auto_Login is false, this parameter is ignored. The default value is obtained from the Account switch in the user's session switch file.

```
Remote_Directory : String := Ftp_Profile.Remote_Directory;
```

Specifies the remote directory context for this transfer. The value "" means that the Connect procedure does not explicitly set the directory. The default is the null string ("").

```
Remote_Roof : String := Ftp_Profile.Remote_Roof;
```

Specifies the remote roof directory to be used when transferring to or from the remote machine. The default is to use the value specified in the Account switch found in the session switch file; a value of "" indicates the current working directory. The default is the null string ("").

```
Remote_Type : Ftp_Name_Map.Machine_Type := Ftp_Profile.Remote_Type;
```

Specifies the computer type for the remote machine. The default is Rational.

```
Transfer_Type : Ftp_Defs.Type.Code := Ftp_Profile.Transfer_Type;
```

Specifies the data representation (Text, Image, Binary) for the transfer. The default is Ascii.

```
Transfer_Mode : Ftp_Defs.Mode_Code := Ftp_Profile.Transfer_Mode;
```

Specifies the mode of transfer. Only Stream (the default) is currently supported.

```
Transfer_Structure : Ftp_Defs.Structure_Code :=
                                 Ftp_Profile.Transfer_Structure;
```

Specifies structure of transfer (File, Recrd, Page). Only File (the default) is currently supported.

```
Send_Port : Boolean := Ftp_Profile.Send_Port_Enabled;
```

Specifies whether the server should be explicitly informed of the port to use for data transfer. It is recommended that this parameter be set to true (the default) when transferring multiple files in a single session.

```
Response : Profile.Response_Profile := Profile.Get;
```

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

Restrictions

Each user session can have only one active FTP connection at a given time. If a second connection is requested without first disconnecting the first connection, the user receives an error message and the new connection will not be formed.

References

procedure Disconnect

procedure Login

procedure Send_Port

---

function Current_Connection

```
function Current_Connection return File_Transfer.Connect_Id;
```

Description

Returns the connection identification number for the current connection.

Parameters

```
return File_Transfer.Connect_Id;
```

Returns the Connect_Id for the current connection.

renamed function Current_Remote_Roof

```
function Current_Remote_Roof return String
                            renames Ftp_Profile.Current_Remote_Roof;
```

Description

Returns the current roof value for the remote host.

This value is used when moving multiple files to or from the remote host to indicate the outermost point in the directory structure.

Parameters

`return String;`

Returns the roof value as a string.

References

procedure Retrieve_Set

procedure Store_Set

procedure Use_Remote_Roof

package Ftp_Name_Map

renamed function Current_Remote_Type

```
function Current_Remote_Type return Ftp_Name_Map.Machine_Type
                             renames Ftp_Profile.Current_Remote_Type;
```

Description

Returns a value indicating the type of machine currently specified for the remote host.

This information is used during automatic generation of remote filenames to determine any translations needed for pathname characters marking directory structure information.

Parameters

`return Ftp_Name_Map.Machine_Type;`

Returns the machine type.

References

procedure Retrieve_Set

procedure Store_Set

procedure Use_Remote_Type

package Ftp_Name_Map

renamed procedure Cwd

```
procedure Cwd
(Remote_Directory  : String                     :=
                                                  Ftp_Profile.Remote_Directory;
 Response          : Profile.Response_Profile  := Profile.Get;
 Account           : String                     := Ftp_Profile.Account)
                                         renames Ftp.Change_Working_Directory;
```

Description

Sets the user's working context on the remote machine to be that specified by the Remote_Directory parameter.

Parameters

`Remote_Directory : String := Ftp_Profile.Remote_Directory;`

Specifies the new directory context. The default is the null string ("").

`Response : Profile.Response_Profile := Profile.Get;`

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

`Account : String := Ftp_Profile.Account;`

Specifies the account information supplied to the remote machine when it requires account information. The default value is obtained from the Account switch in the user's session switch file.

Restrictions

The user first must have connected and logged into a remote server before issuing this command. If the session does not have an active FTP connection, an error message is produced and the command has no effect.

References

procedure Change_Working_Directory

package Ftp_Profile

procedure Delete

```
procedure Delete
           (Remote_File : String;
            Response    : Profile.Response_Profile := Profile.Get;
            Account     : String                   := Ftp_Profile.Account);
```

Description

Sends a request for the remote host to delete a file.

Parameters

`Remote_File : String;`

Specifies a simple filename for a file to be deleted on the remote host.

`Response : Profile.Response_Profile := Profile.Get;`

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

`Account : String := Ftp_Profile.Account;`

Specifies the account information supplied to the remote machine when it requires account information. The default value is obtained from the Account switch in the user's session switch file.

Restrictions

The user first must have connected and logged into a remote server before issuing this command. If the session does not have an active FTP connection, an error message is produced and the command has no effect.

procedure Disconnect

```
procedure Disconnect (Response : Profile.Response_Profile := Profile.Get);
```

Description

Disconnects the current session from the remote machine.

This procedure sends the FTP protocol "QUIT" command and then disconnects the network connection, freeing local resources.

If the user is not connected, nothing is done and an error message is written into the log file.

Parameters

```
Response : Profile.Response_Profile := Profile.Get;
```

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

References

procedure Abandon

procedure Connect

constant Ebcdic

```
Ebcdic : constant Ftp_Defs.Type_Code := Ftp_Defs.Ebcdic;
```

Description

Defines a constant that imports the Type_Code value of Ebcdic from Ftp_Defs.

This constant indicates that the file being transferred uses the eight-bit EBCDIC character representation.

constant Ebcdic_Cc

```
Ebcdic_Cc : constant Ftp_Defs.Type_Code := Ftp_Defs.Ebcdic_Cc;
```

Description

Defines a constant that imports the Type_Code value of Ebcdic_Telnet from Ftp-_Defs.

This constant indicates that the file being transferred uses the eight-bit EBCDIC character representation with ASA (FORTRAN) vertical control.

constant Ebcdic_Telnet

```
Ebcdic_Telnet : constant Ftp_Defs.Type_Code := Ftp_Defs.Ebcdic_Telnet
```

Description

Defines a constant that imports the Type_Code value of Ebcdic_Telnet from Ftp-_Defs.

This constant indicates that the file being transferred uses the eight-bit EBCDIC character representation with Telnet vertical control.

constant File

```
File : constant Ftp_Defs.Structure_Code := Ftp_Defs.File;
```

Description

Defines a constant that imports the Structure_Code value of File from Ftp_Defs.

This constant indicates that the file being transmitted is a continuous sequence of data bytes.

procedure Get

```
procedure Get
(From_Remote_File    : String                      := "";
 To_Local_File       : String                      := "";
 Remote_Machine      : String                      :=
                                                   Ftp_Profile.Remote_Machine;
 Username            : String                      := Ftp_Profile.Username;
 Password            : String                      := Ftp_Profile.Password;
 Account             : String                      := Ftp_Profile.Account;
 Remote_Directory    : String                      :=
                                                 Ftp_Profile.Remote_Directory;
 Remote_Type         : Ftp_Name_Map.Machine_Type   := Ftp_Profile.Remote_Type;
 Append_To_File      : Boolean                     := False;
 Transfer_Type       : Ftp_Defs.Type_Code          :=
                                                   Ftp_Profile.Transfer_Type;
 Transfer_Mode       : Ftp_Defs.Mode_Code          :=
                                                   Ftp_Profile.Transfer_Mode;
 Transfer_Structure  : Ftp_Defs.Structure_Code     :=
                                               Ftp_Profile.Transfer_Structure;
 Send_Port           : Boolean                     :=
                                                Ftp_Profile.Send_Port_Enabled;
 Response            : Profile.Response_Profile    := Ftp.Profile_Get);
```

Description

Retrieves a file from a remote host.

This procedure is used as a one-step operation to retrieve a file from a remote host. It forms a connection that logs in the user and retrieves the file. At the end of the transfer, the connection is disconnected.

Parameters

```
From_Remote_File  :  String := "";
```

Specifies the file to be retrieved from the remote host. A filename must be given for this parameter.

```
To_Local_File  :  String := "";
```

Specifies the local file into which the copy is to be written. If the null string is given, then a local name is built using the remote name and the Ftp_Name_Map-.Remote_To_Local function.

```
Remote_Machine  :  String := Ftp_Profile.Remote_Machine;
```

Specifies the remote machine to which the connection should be formed.

```
Username : String := Ftp_Profile.Username;
```

Specifies the username to be supplied to the remote server for login.

```
Password : String := Ftp_Profile.Password;
```

Specifies the password to be supplied to the remote server for login.

```
Account : String := Ftp_Profile.Account;
```

Specifies the account information supplied to the remote machine when it requires account information. The default value is obtained from the Account switch in the user's session switch file.

```
Remote_Directory : String := Ftp_Profile.Remote_Directory;
```

Specifies the remote directory context for this transfer. The value "" means that the Get procedure does not explicitly set the directory.

```
Remote_Type : Ftp_Name_Map.Machine_Type := Ftp_Profile.Remote_Type;
```

Specifies the machine type of the remote machine. This information can be used when forming the local filename. The default is Rational.

```
Append_To_File : Boolean := False;
```

Specifies, if true, that the retrieved file should be appended to the local file if it already exists. If the file does not exist, it is created. If false, the local file is overwritten if it exists and is created if it does not exist.

```
Transfer_Type : Ftp_Defs.Type_Code := Ftp_Profile.Transfer_Type;
```

Specifies the representation type to be used while transferring the file. The default is Ascii.

```
Transfer_Mode : Ftp_Defs.Mode_Code := Ftp_Profile.Transfer_Mode;
```

Specifies the transfer mode to be used while transferring the file. The default is Stream.

```
Transfer_Structure : Ftp_Defs.Structure_Code :=
                                        Ftp_Profile.Transfer_Structure;
```

Specifies the transfer structure to be used while transferring the file. The default is File.

```
Send_Port : Boolean := Ftp_Profile.Send_Port_Enabled;
```

Specifies whether the server should be explicitly informed of the port to use for data transfer. It is recommended that this parameter be set to true (the default) when transferring multiple files in a single session.

```
Response : Profile.Response_Profile := Ftp.Profile_Get;
```

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

Restrictions

If a connection already exists for the session, the transfer does not occur and an error message is generated.

Errors

If any of the following occur:

- The connection cannot be formed
- The user cannot log in
- The specified transfer parameters are not supported
- Moving to the specified remote directory fails

the command will fail and no files will be transferred.

References

function Get_Set

function Put

function Put_Set

procedure Send_Port

package Ftp_Defs

package Ftp_Name_Map

function Ftp_Name_Map.Remote_To_Local

package Ftp_Profile

---

procedure Get_List

```
procedure Get_List
(Remote_File_List     : String                        := "";
 Local_Roof           : String                        := "$";
 Remote_Roof          : String                        := Ftp_Profile.Remote_Roof;
 Remote_Machine       : String                        :=
                                                     Ftp_Profile.Remote_Machine;
 Username             : String                        := Ftp_Profile.Username;
 Password             : String                        := Ftp_Profile.Password;
 Account              : String                        := Ftp_Profile.Account;
 Remote_Directory     : String                        :=
                                                   Ftp_Profile.Remote_Directory;
 Remote_Type          : Ftp_Name_Map.Machine_Type     := Ftp_Profile.Remote_Type;
 Append_To_File       : Boolean                       := False;
 Transfer_Type        : Ftp_Defs.Type_Code            :=
                                                      Ftp_Profile.Transfer_Type;
 Transfer_Mode        : Ftp_Defs.Mode_Code            :=
                                                      Ftp_Profile.Transfer_Mode;
 Transfer_Structure   : Ftp_Defs.Structure_Code       :=
                                                 Ftp_Profile.Transfer_Structure;
 Send_Port            : Boolean                       :=
                                                  Ftp_Profile.Send_Port_Enabled;
 Response             : Profile.Response_Profile      := Profile.Get);
```

Description

Retrieves multiple files from a remote host.

This procedure is used as a one-step operation to retrieve multiple files from a remote host. A connection is formed to the specified machine. The connection is logged in using the user information supplied. Files specified in the Remote_File_List parameter are retrieved. At the end of the transfer, the connection is disconnected.

The names of the local files are generated algorithmically from the names of the remote files, using the Ftp_Name_Map.Remote_To_Local function. The Local-_Roof, Remote_Roof, and Remote_Type parameters are used in this conversion.

Parameters

`Remote_File_List : String := "";`

Specifies the name of a local file. The file must contain text, with each line containing the name of one remote file.

`Local_Roof : String := "$";`

Specifies the parent for the file(s) to be retrieved.

```
Remote_Roof : String := Ftp_Profile.Remote_Roof;
```

Specifies the parent on the remote machine for all files to be retrieved. If the null string is given, retrieved filenames are flattened; that is, the simple names of the source files are mapped to the simple names of the target files.

```
Remote_Machine : String := Ftp_Profile.Remote_Machine;
```

Specifies the machine to which the connection is formed and from which files are retrieved.

```
Username : String := Ftp_Profile.Username;
```

Specifies the username to be supplied to the remote server for login.

```
Password : String := Ftp_Profile.Password;
```

Specifies the password to be supplied to the remote server for login.

```
Account : String := Ftp_Profile.Account;
```

Specifies the account to be supplied to the remote server, if required.

```
Remote_Directory : String := Ftp_Profile.Remote_Directory;
```

Specifies the remote directory context for this transfer. The value "" means that the Get_List procedure does not explicitly set the directory.

```
Remote_Type : Ftp_Name_Map.Machine_Type := Ftp_Profile.Remote_Type;
```

Specifies the machine type of the remote machine. This information is used when forming local filenames. The default is Rational.

```
Append_To_File : Boolean := False;
```

Specifies, if true, that each retrieved file should be appended to the local file if it already exists. If the file does not exist, it is created. If false, the local file is overwritten if it exists and is created if it does not exist.

```
Transfer_Type : Ftp_Defs.Type_Code := Ftp_Profile.Transfer_Type;
```

Specifies the representation type to be used while transferring the files. The default is Ascii.

```
Transfer_Mode : Ftp_Defs.Mode_Code := Ftp_Profile.Transfer_Mode;
```

Specifies the transfer mode to be used while transferring the files. The default is Stream.

```
Transfer_Structure  :  Ftp_Defs.Structure_Code  :=
                                       Ftp_Profile.Transfer_Structure;
```

Specifies the transfer structure to be used while transferring the files. The default is File.

```
Send_Port  :  Boolean := Ftp_Profile.Send_Port_Enabled;
```

Specifies whether the server should be explicitly informed of the port to use for data transfer before starting the transfer. It is recommended that this parameter be set to true (the default) when transferring multiple files in a single session.

```
Response  :  Profile.Response_Profile  := Profile.Get;
```

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

Restrictions

If a connection already exists for the session, the transfer does not occur and an error message is generated.

Errors

If the Remote_File_List cannot be opened or does not contain valid remote filenames, this command will not work.

If any of the following occur:

- The connection cannot be formed
- The user cannot log in
- The specified transfer parameters are not supported
- Moving to the specified remote directory fails

the command will fail and no files will be transferred.

If a file fails to transfer, the command may attempt to transfer the remaining files or terminate, depending on the specified Response_Profile (Profile.Reaction (Response)).

References

procedure Get

procedure Get_Set

procedure Retrieve_List

procedure Send_Port

function Ftp_Name_Map.Remote_To_Local

procedure Get_Set

```
procedure Get_Set
(From_Remote_File_Set  : String                       := "";
 Local_Roof            : String                       := "$";
 Remote_Roof           : String                       :=
                                                        Ftp_Profile.Remote_Roof;
 Remote_Machine        : String                       :=
                                                     Ftp_Profile.Remote_Machine;
 Username              : String                       := Ftp_Profile.Username;
 Password              : String                       := Ftp_Profile.Password;
 Account               : String                       := Ftp_Profile.Account;
 Remote_Directory      : String                       :=
                                                   Ftp_Profile.Remote_Directory;
 Remote_Type           : Ftp_Name_Map.Machine_Type    :=
                                                        Ftp_Profile.Remote_Type;
 Append_To_File        : Boolean                      := False;
 Transfer_Type         : Ftp_Defs.Type_Code           :=
                                                      Ftp_Profile.Transfer_Type;
 Transfer_Mode         : Ftp_Defs.Mode_Code           :=
                                                      Ftp_Profile.Transfer_Mode;
 Transfer_Structure    : Ftp_Defs.Structure_Code      :=
                                                 Ftp_Profile.Transfer_Structure;
 Send_Port             : Boolean                      :=
                                                  Ftp_Profile.Send_Port_Enabled;
 Response              : Profile.Response_Profile     := Profile.Get);
```

Description

Retrieves multiple files from a remote host.

This procedure is used as a one-step operation to retrieve multiple files from a remote host. A connection is formed to the specified machine. The connection is logged in using the user information supplied. Files matching the template From_Remote_File_Set are retrieved and placed in the context indicated by the Local_Roof parameter. At the end of the transfer, the connection is disconnected.

The names of the local files are generated algorithmically from the names of the remote files, using the Ftp_Name_Map.Remote_To_Local function. The Local_Roof, Remote_Roof, and Remote_Type parameters are used in this conversion.

Parameters

`From_Remote_File_Set  :  String := "";`

Specifies a template for the files to be retrieved from the remote host. The effect of the null string may vary for each implementation of the remote server. It is recommended that a nonnull argument be given. The wildcards used must be those of the remote machine.

```
Local_Roof : String := "$";
```

Specifies the parent for the file(s) to be retrieved.

```
Remote_Roof : String := Ftp_Profile.Remote_Roof;
```

Specifies the parent on the remote machine for all files to be retrieved. If the null string is given, retrieved filenames are flattened; that is, the simple names of the source files are mapped to the simple names of the target files.

```
Remote_Machine : String := Ftp_Profile.Remote_Machine;
```

Specifies the machine to which the connection is formed and from which files are retrieved.

```
Username : String := Ftp_Profile.Username;
```

Specifies the username to be supplied to the remote server for login.

```
Password : String := Ftp_Profile.Password;
```

Specifies the password to be supplied to the remote server for login.

```
Account : String := Ftp_Profile.Account;
```

Specifies the account information supplied to the remote machine when it requires account information. The default value is obtained from the Account switch in the user's session switch file.

```
Remote_Directory : String := Ftp_Profile.Remote_Directory;
```

Specifies the remote directory context for this transfer. The value "" means that the Get_Set procedure does not explicitly set the directory.

```
Remote_Type : Ftp_Name_Map.Machine_Type := Ftp_Profile.Remote_Type;
```

Specifies the machine type of the remote machine. This information is used when forming local filenames. The default is Rational.

```
Append_To_File : Boolean := False;
```

Specifies, if true, that the retrieved file should be appended to the local file if it already exists. If the file does not exist, it is created. If false, the local file is overwritten if it exists and is created if it does not exist.

```
Transfer_Type : Ftp_Defs.Type_Code := Ftp_Profile.Transfer_Type;
```

Specifies the representation type to be used while transferring the files. The default is Ascii.

```
Transfer_Mode : Ftp_Defs.Mode_Code := Ftp_Profile.Transfer_Mode;
```

Specifies the transfer mode to be used while transferring the files. The default is Stream.

```
Transfer_Structure : Ftp_Defs.Structure_Code :=
                                        Ftp_Profile.Transfer_Structure;
```

Specifies the transfer structure to be used while transferring the files. The default is File.

```
Send_Port : Boolean := Ftp_Profile.Send_Port_Enabled;
```

Specifies whether the server should be explicitly informed of the port to use for data transfer. It is recommended that this parameter be set to true (the default) when transferring multiple files in a single session.

```
Response : Profile.Response_Profile := Profile.Get;
```

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

Restrictions

If a connection already exists for the session, the transfer does not occur and an error message is generated.

Errors

The list of files to be retrieved is generated by requesting the remote server to produce a list of all files matching the specified template. If the remote server does not support this function, the Get_Set procedure will not work.

If any of the following occur:

- The connection cannot be formed
- The user cannot log in
- The specified transfer parameters are not supported
- Moving to the specified remote directory fails

the command will fail and no files will be transferred.

If a file fails to transfer, the command may attempt to transfer the remaining files or terminate, depending on the specified Response_Profile (Profile.Reaction (Response)).

References

procedure Get

procedure Put

procedure Put_Set

procedure Send_Port

package Ftp_Defs

package Ftp_Name_Map

function Ftp_Name_Map.Remote_To_Local

package Ftp_Profile

constant Image

```
Image : constant Ftp_Defs.Type_Code := Ftp_Defs.Image;
```

Description

Defines a constant that imports the Type_Code value of Image from Ftp_Defs.

This constant indicates that binary data are transferred as contiguous bits of eight-bit transfer bytes.

procedure List

```
procedure List
       (Remote_Pathname  : String                   := "";
        Verbose          : Boolean                  := True;
        To_Local_File    : String                   := "";
        Response         : Profile.Response_Profile := Profile.Get;
        Account          : String                   := Ftp_Profile.Account);
```

Description

Produces a directory listing from the remote host.

Parameters

`Remote_Pathname : String := "";`

Specifies a name or template for which a directory listing is to be produced.

`Verbose : Boolean := True;`

Creates, if true, a verbose listing containing file information such as size. If false, only the names of the files are listed, one per line.

`To_Local_File : String := "";`

Specifies the destination for the file listing. The null string specifies that the listing is to be routed to the log file.

`Response : Profile.Response_Profile := Profile.Get;`

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

`Account : String := Ftp_Profile.Account;`

Specifies the account information supplied to the remote machine when it requires account information. The default value is obtained from the Account switch found in the user's session switch file.

Restrictions

The user first must have connected and logged into a remote server before issuing this command. If the session does not have an active FTP connection, an error message is produced and the command has no effect.

References

procedure Remote_Status

---

constant Local_Binary

```
Local_Binary : constant Ftp_Defs.Type_Code := Ftp_Defs.Local_Binary;
```

Description

Defines a constant that imports the Type_Code value of Local_Binary from Ftp-_Defs.

This constant indicates that the data transferred are not byte-aligned or are from a server that does not support binary mode.

constant Local_Byte

```
Local_Byte : constant Ftp_Defs.Type_Code := Ftp_Defs.Local_Byte;
```

Description

Defines a constant that imports the Type_Code value of Local_Byte from Ftp_Defs.

This constant indicates that the data are transferred in logical bytes of the size specified with the type.

procedure Login

```
procedure Login
(Username             : String                     := Ftp_Profile.Username;
 Password             : String                     := Ftp_Profile.Password;
 Account              : String                     := Ftp_Profile.Account;
 Remote_Directory     : String                     :=
                                                 Ftp_Profile.Remote_Directory;
 Remote_Roof          : String                     := Ftp_Profile.Remote_Roof;
 Remote_Type          : Ftp_Name_Map.Machine_Type  := Ftp_Profile.Remote_Type;
 Transfer_Type        : Ftp_Defs.Type.Code         :=
                                                   Ftp_Profile.Transfer_Type;
 Transfer_Mode        : Ftp_Defs.Mode_Code         :=
                                                   Ftp_Profile.Transfer_Mode;
 Transfer_Structure   : Ftp_Defs.Structure_Code    :=
                                              Ftp_Profile.Transfer_Structure;
 Send_Port            : Boolean                    :=
                                               Ftp_Profile.Send_Port_Enabled;
 Response             : Profile.Response_Profile   := Profile.Get);
```

Description

Performs login procedures for the current connection.

This command is used when the Auto_Login feature of the Connect procedure is not used.

Parameters

`Username : String := Ftp_Profile.Username;`

Specifies the username for the remote host.

`Password : String := Ftp_Profile.Password;`

Specifies the password for the remote host.

`Account : String := Ftp_Profile.Account;`

Specifies the account information supplied to the remote machine when it requires account information. The default value is obtained from the Account switch in the user's session switch file.

`Remote_Directory : String := Ftp_Profile.Remote_Directory;`

Specifies the remote directory context for this transfer. The value "" means that the Login procedure does not explicitly set the directory.

```
Remote_Roof : String := Ftp_Profile.Remote_Roof;
```

Specifies the remote roof directory to be used when transferring to or from the remote machine. The default is to use the value specified in the Remote_Roof switch found in the session switch file; a value of "" indicates the current working directory.

```
Remote_Type : Ftp_Name_Map.Machine_Type := Ftp_Profile.Remote_Type;
```

Specifies the computer type for the remote machine. The default is Rational.

```
Transfer_Type : Ftp_Defs.Type.Code := Ftp_Profile.Transfer_Type;
```

Specifies the data representation (Text, Image, Binary) for transfer. The default is Ascii.

```
Transfer_Mode : Ftp_Defs.Mode_Code := Ftp_Profile.Transfer_Mode;
```

Specifies the mode of transfer. Only Stream (the default) is currently supported.

```
Transfer_Structure : Ftp_Defs.Structure_Code :=
                                     Ftp_Profile.Transfer_Structure;
```

Specifies the structure of transfer (File, Recrd, Page). Only File (the default) is currently supported.

```
Send_Port : Boolean := Ftp_Profile.Send_Port_Enabled;
```

Specifies whether the server should be explicitly informed of the port to use for data transfer. It is recommended that this parameter be set to true (the default) when transferring multiple files in a single session.

```
Response : Profile.Response_Profile := Profile.Get;
```

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

Restrictions

A connection to a remote host must already exist for this command to work. If a connection does not exist, this command has no effect and an error message is generated.

Example

```
Ftp.Connect (To_Machine => "The_Machine",
             Auto_Login => False);
Ftp.Login (Username => "Myusername",
           Password => "Mypassword");
```

References

procedure Connect

procedure Send_Port

package Ftp_Profile

constant Mv

```
Mv : constant Ftp_Name_Map.Machine_Type := Aos;
```

Description

Defines a constant that imports a Machine_Type value of Aos from Ftp_Name_Map.

This constant is used to indicate that the transfer is to/from a Data General computer using the AOS operating system.

constant Mvs

```
Mvs : constant Ftp_Name_Map.Machine_Type  := Ftp_Name_Map.Mvs;
```

Description

Defines a constant that imports the Machine_Type value of Mvs from Ftp_Name_Map.

This constant is used to indicate that the transfer is to/from an IBM computer using the MVS operating system. It is not supported.

constant Page

```
Page : constant Ftp_Defs.Structure_Code  := Ftp_Defs.Page;
```

Description

Defines a constant that imports the Structure_Code value of Page from Ftp_Defs.

This constant is used to indicate that the file consists of independent, indexed pages.

renamed function Profile_Get

```
function Profile_Get return Profile.Response_Profile  renames Profile.Get;
```

Description

Returns the current response profile for the current job.

This function is used as a parameter to commands in the Environment to supply the current response profile to those commands. The function returns the response profile for the current job. If a profile for the job has not been defined, the session default profile is returned.

Parameters

```
return Profile.Response_Profile;
```

Returns the job response profile. If no job response profile has been explicitly defined, the session response profile is returned.

procedure Put

```
procedure Put
(From_Local_File     : String                        := "<IMAGE>";
 To_Remote_File      : String                        := "";
 Remote_Machine      : String                        :=
                                                     Ftp_Profile.Remote_Machine;
 Username            : String                        := Ftp_Profile.Username;
 Password            : String                        := Ftp_Profile.Password;
 Account             : String                        := Ftp_Profile.Account;
 Remote_Directory    : String                        :=
                                                   Ftp_Profile.Remote_Directory;
 Remote_Type         : Ftp_Name_Map.Machine_Type     := Ftp_Profile.Remote_Type;
 Append_To_File      : Boolean                       := False;
 Transfer_Type       : Ftp_Defs.Type_Code            :=
                                                      Ftp_Profile.Transfer_Type;
 Transfer_Mode       : Ftp_Defs.Mode_Code            :=
                                                      Ftp_Profile.Transfer_Mode;
 Transfer_Structure  : Ftp_Defs.Structure_Code       :=
                                                 Ftp_Profile.Transfer_Structure;
 Send_Port           : Boolean                       :=
                                                  Ftp_Profile.Send_Port_Enabled;
 Response            : Profile.Response_Profile      := Profile.Get);
```

Description

Stores a file onto a remote host.

This procedure is used as a one-step operation to store a file onto a remote host. A connection is formed to the specified machine. The connection is logged in using the user information supplied. The local file is transferred to the remote machine and placed in the remote file. At the end of the transfer, the connection is disconnected.

Parameters

`From_Local_File : String := "<IMAGE>";`

Specifies the local file to be transferred to the remote machine. Its value should not be the null string. The default, "<IMAGE>", references the object associated with the highlighted area in which the cursor is located. If the cursor is not located in the highlighted area, "<IMAGE>" references the object associated with the window in which the cursor is located.

`To_Remote_File : String := "";`

Specifies the destination filename. If null, the filename is derived from the local filename, using the Ftp_Name_Map.Local_To_Remote function.

```
Remote_Machine : String := Ftp_Profile.Remote_Machine;
```

Specifies the machine to which the connection is formed and to which the file is stored. The default is the null string ("").

```
Username : String := Ftp_Profile.Username;
```

Specifies the username to be supplied to the remote server for login.

```
Password : String := Ftp_Profile.Password;
```

Specifies the password to be supplied to the remote server for login. The default is the null string ("").

```
Account : String := Ftp_Profile.Account;
```

Specifies the account to be supplied to the remote server, if required.

```
Remote_Directory : String := Ftp_Profile.Remote_Directory;
```

Specifies the remote directory context for this transfer. The value "" means that the Put procedure does not explicitly set the directory. The default is the null string ("").

```
Remote_Type : Ftp_Name_Map.Machine_Type := Ftp_Profile.Remote_Type;
```

Specifies the machine type of the remote machine. This information is used when forming remote filenames. The default is Rational.

```
Append_To_File : Boolean := False;
```

Specifies, if true, that the stored file should be appended to the remote file if it already exists. If the file does not exist, it is created. If false, the remote file is overwritten if it already exists and is created if it does not exist.

```
Transfer_Type : Ftp_Defs.Type_Code := Ftp_Profile.Transfer_Type;
```

Specifies the representation type to be used while transferring the file. The default is Ascii.

```
Transfer_Mode : Ftp_Defs.Mode_Code := Ftp_Profile.Transfer_Mode;
```

Specifies the transfer mode to be used while transferring the file. The default is Stream.

```
Transfer_Structure : Ftp_Defs.Structure_Code :=
                                   Ftp_Profile.Transfer_Structure;
```

Specifies the transfer structure to be used while transferring the file. The default is File.

```
Send_Port  :  Boolean := Ftp_Profile.Send_Port_Enabled;
```

Specifies whether the server should be explicitly informed of the port to use for data transfer. It is recommended that this parameter be set to true (the default) when transferring multiple files in a single session.

```
Response  :  Profile.Response_Profile  := Profile.Get;
```

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

Restrictions

If a connection already exists for the session, the transfer does not occur and an error message is generated.

Errors

If any of the following occur:

- The connection cannot be formed
- The user cannot log in
- The specified transfer parameters are not supported
- Moving to the specified remote directory fails

the command will fail and no files will be transferred.

References

procedure Get

procedure Put_Set

procedure Send_Port

function Ftp_Name_Map.Local_To_Remote

---

procedure Put_Set

```
procedure Put_Set
(From_Local_File_Set  : String                       := "<IMAGE>";
 Local_Roof           : String                       := "$";
 Remote_Roof          : String                       :=
                                                         Ftp_Profile.Remote_Roof;
 Remote_Machine       : String                       :=
                                                      Ftp_Profile.Remote_Machine;
 Username             : String                        := Ftp_Profile.Username;
 Password             : String                        := Ftp_Profile.Password;
 Account              : String                        := Ftp_Profile.Account;
 Remote_Directory     : String                       :=
                                                    Ftp_Profile.Remote_Directory;
 Remote_Type          : Ftp_Name_Map.Machine_Type    :=
                                                         Ftp_Profile.Remote_Type;
 Append_To_File       : Boolean                      := False;
 Transfer_Type        : Ftp_Defs.Type_Code           :=
                                                       Ftp_Profile.Transfer_Type;
 Transfer_Mode        : Ftp_Defs.Mode_Code           :=
                                                       Ftp_Profile.Transfer_Mode;
 Transfer_Structure   : Ftp_Defs.Structure_Code      :=
                                                  Ftp_Profile.Transfer_Structure;
 Send_Port            : Boolean                      :=
                                                   Ftp_Profile.Send_Port_Enabled;
 Response             : Profile.Response_Profile     := Profile.Get);
```

Description

Stores multiple files onto a remote host.

This procedure is used as a one-step operation to store multiple files onto a remote host. A connection is formed to the specified machine. The connection is logged in using the user information supplied. Files matching the template for the From_Local_File_Set parameter are stored in the context indicated by the Remote_Roof parameter. At the end of the transfer, the connection is disconnected.

Parameters

`From_Local_File : String := "<IMAGE>";`

Specifies the local files to be transferred to the remote machine. It should be a nonnull argument. The default, "<IMAGE>", references the object associated with the highlighted area in which the cursor is located. If the cursor is not located in the highlighted area, "<IMAGE>" references the object associated with the window in which the cursor is located.

```
Local_Roof : String := "$";
```

Specifies the ancestor directory for the file(s) to be transferred. If the null string is given, retrieved filenames are flattened; that is, the simple names of the source files are mapped to the simple names of the target files.

```
Remote_Roof : String := Ftp_Profile.Remote_Roof;
```

Specifies the ancestor directory on the remote machine for all files stored. The default is the null string ("").

```
Remote_Machine : String := Ftp_Profile.Remote_Machine;
```

Specifies the machine to which the connection is formed and to which files are transferred. The default is the null string ("").

```
Username : String := Ftp_Profile.Username;
```

Specifies the username to be supplied to the remote server for login.

```
Password : String := Ftp_Profile.Password;
```

Specifies the password to be supplied to the remote server for login.

```
Account : String := Ftp_Profile.Account;
```

Specifies the account information supplied to the remote machine when it requires account information. The default value is obtained from the Account switch in the user's session switch file.

```
Remote_Directory : String := Ftp_Profile.Remote_Directory;
```

Specifies the remote directory context for this transfer. The value "" means that the Put_Set procedure does not explicitly set the directory.

```
Remote_Type : Ftp_Name_Map.Machine_Type := Ftp_Profile.Remote_Type;
```

Specifies the machine type of the remote machine. This information is used when forming remote filenames. The default is Rational.

```
Append_To_File : Boolean := False;
```

Specifies, if true, that the transferred file should be appended to the destination file if it already exists. If the file does not exist, it is created. If false, the destination file is overwritten if it exists and is created if it does not exist.

```
Transfer_Type : Ftp_Defs.Type_Code := Ftp_Profile.Transfer_Type;
```

Specifies the representation type to be used while transferring the files. The default is Ascii.

```
Transfer_Mode : Ftp_Defs.Mode_Code := Ftp_Profile.Transfer_Mode;
```

Specifies the transfer mode to be used while transferring the files. The default is Stream.

```
Transfer_Structure : Ftp_Defs.Structure_Code :=
                                      Ftp_Profile.Transfer_Structure;
```

Specifies the transfer structure to be used while transferring the files. The default is File.

```
Send_Port : Boolean := Ftp_Profile.Send_Port_Enabled;
```

Specifies whether the server should be explicitly informed of the port to use for data transfer. It is recommended that this parameter be set to true (the default) when transferring multiple files in a single session.

```
Response : Profile.Response_Profile := Profile.Get;
```

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

Restrictions

If a connection already exists for the session, the transfers do not occur and an error message is generated.

Errors

If any of the following occur:

- The connection cannot be formed
- The user cannot log in
- The specified transfer parameters are not supported
- Moving to the specified remote directory fails

the command will fail and no files will be transferred.

If a file fails to transfer, the command may attempt to transfer the remaining files or terminate, depending on the specified Response_Profile (Profile.Reaction (Response)).

References

procedure Get

procedure Get_Set

procedure Put

procedure Send_Port

package Ftp_Defs

package Ftp_Name_Map

package Ftp_Profile

constant Rational

```
Rational : constant Ftp_Name_Map.Machine_Type := Ftp_Name_Map.Rational;
```

Description

Defines a constant that imports the Machine_Type value of Rational from Ftp-_Name_Map.

This constant is used to denote a machine running the Rational Environment.

constant R1000

```
R1ØØØ : constant Ftp_Name_Map.Machine_Type  := Ftp_Name_Map.Rational;
```

Description

Defines a constant that imports the Machine_Type value of R1000 from Ftp_Name_Map.

This constant is used to denote a machine running the Rational Environment.

constant Recrd

```
Recrd : constant Ftp_Defs.Structure_Code  := Ftp_Defs.Recrd;
```

Description

Defines a constant that imports the Structure_Code value of Recrd from Ftp_Defs.

This constant is used to indicate that the file consists of sequential records.

procedure Remote_Help

```
procedure Remote_Help
               (Argument  : String                    := "";
                Response  : Profile.Response_Profile  := Profile.Get;
                Account   : String                    := Ftp_Profile.Account);
```

Description

Requests help information from the remote server for commands.

Parameters

`Argument  :  String := "";`

Specifies the optional argument to the help command.

`Response  :  Profile.Response_Profile  := Profile.Get;`

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

`Account  :  String := Ftp_Profile.Account;`

Specifies the account information supplied to the remote machine when it requires account information. The default value is obtained from the Account switch in the user's session switch file.

Restrictions

The user first must have connected and logged into a remote server before issuing this command. If the session does not have an active FTP connection, an error message is produced and the command has no effect.

procedure Remote_Status

```
procedure Remote_Status
            (Argument : String                   := "";
             Response : Profile.Response_Profile := Profile.Get;
             Account  : String                   := Ftp_Profile.Account);
```

Description

Requests status information from the remote server.

The information returned varies from machine to machine. If an argument is supplied, directory/file information is returned for the file(s) specified by the argument.

Parameters

`Argument : String := "";`

Specifies the optional argument.

`Response : Profile.Response_Profile := Profile.Get;`

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

`Account : String := Ftp_Profile.Account;`

Specifies the account information supplied to the remote machine when it requires account information. The default value is obtained from the Account switch in the user's session switch file.

Restrictions

The user first must have connected and logged into a remote server before issuing this command. If the session does not have an active FTP connection, an error message is produced and the command has no effect.

procedure Retrieve

```
procedure Retrieve
  (From_Remote_File  : String                        := "";
   To_Local_File     : String                        := "";
   Append_To_File    : Boolean                       := False;
   Response          : Profile.Response_Profile      := Profile.Get;
   Remote_Type       : Ftp_Name_Map.Machine_Type     := Ftp.Current_Remote_Type;
   Account           : String                        := Ftp_Profile.Account);
```

Description

Retrieves a copy of a file from a remote machine.

The user session must already be connected and logged into the remote machine.

Parameters

`From_Remote_File : String := "";`

Specifies the remote file to be copied.

`To_Local_File : String := "";`

Specifies the local file into which the copy is to be written. If the null string is specified, the file is derived from the remote filename, using the Ftp_Name_Map.Remote_To_Local function.

`Append_To_File : Boolean := False;`

Specifies, if true, that the retrieved file should be appended to the local file if it already exists. If the file does not exist, it is created. If false, the local file is overwritten if it already exists and is created if it does not exist.

`Response : Profile.Response_Profile := Profile.Get;`

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

`Remote_Type : Ftp_Name_Map.Machine_Type := Ftp.Current_Remote_Type;`

Specifies the machine type of the remote machine. This information is used in generating the local filenames, because some differences may exist in directory-naming and filenaming conventions. The default is Rational.

```
Account  :  String := Ftp_Profile.Account;
```

Specifies the account information supplied to the remote machine when it requires account information. The default value is obtained from the Account switch in the user's session switch file.

Restrictions

The user first must have connected and logged into a remote server before issuing this command. If the session does not have an active FTP connection, an error message is produced and the command has no effect.

References

procedure Connect

procedure Login

procedure Store

function Ftp_Name_Map.Remote_To_Local

---

procedure Retrieve_List

```
procedure Retrieve_List
   (Remote_File_List : String                       := "";
    Local_Roof       : String                       := "$";
    Remote_Roof      : String                       := Ftp.Current_Remote_Roof;
    Remote_Type      : Ftp_Name_Map.Machine_Type    := Ftp.Current_Remote_Type;
    Append_To_File   : Boolean                      := False;
    Response         : Profile.Response_Profile     := Profile.Get;
    Account          : String                       := Ftp_Profile.Account);
```

Description

Retrieves multiple files from a remote machine, using a list of remote filenames stored in a file on the local machine.

The files are stored on the local machine with names mapped from the remote filenames. The names of the local files are generated algorithmically from the names of the remote files, using the Ftp_Name_Map.Remote_To_Local function. The Local_Roof, Remote_Roof, and Remote_Type parameters are used in this conversion.

Parameters

`Remote_File_List : String := "";`

Specifies the name of a local file. The file must contain text, with each line containing the name of one remote file to be retrieved.

`Local_Roof : String := "$";`

Specifies the local parent of the files to be retrieved.

`Remote_Roof : String := Ftp.Current_Remote_Roof;`

Specifies the parent for the files at the remote machine. If the null string is given, retrieved filenames are flattened; that is, the simple names of the source files are mapped to the simple names of the target files.

`Remote_Type : Ftp_Name_Map.Machine_Type := Ftp.Current_Remote_Type;`

Specifies the machine type of the remote machine. This information is used in generating the local filenames, because some differences may exist in directory-naming and filenaming conventions. The default is Rational.

```
Append_To_File : Boolean := False;
```

Specifies, if true, that retrieved files should be appended to local files that already exist. If the local files do not exist, they are created. If false, the local files are overwritten if they already exist and are created if they do not exist.

```
Response : Profile.Response_Profile := Profile.Get;
```

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

```
Account : String := Ftp_Profile.Account;
```

Specifies the account information supplied to the remote machine when it requires account information. The default value is obtained from the Account switch in the user's session switch file.

Restrictions

The user first must have connected and logged into a remote server before issuing this command. If the session does not have an active FTP connection, an error message is produced and the command has no effect.

Errors

If the Remote_File_List file cannot be opened or does not contain valid remote filenames, this command does not work.

If a file fails to transfer, the command may attempt to transfer the remaining files or terminate, depending on the specified Response_Profile (Profile.Reaction (Response)).

References

procedure Get_List

procedure Retrieve

function Ftp_Name_Map.Remote_To_Local

procedure Retrieve_Set

```
procedure Retrieve_Set
(From_Remote_File_Set  : String                      := "";
 Local_Roof            : String                      := "$";
 Remote_Roof           : String                      :=
                                                        Ftp.Current_Remote_Roof;
 Remote_Type           : Ftp_Name_Map.Machine_Type   :=
                                                        Ftp.Current_Remote_Type;
 Append_To_File        : Boolean                     := False;
 Response              : Profile.Response_Profile    := Profile.Get;
 Account               : String                      := Ftp_Profile.Account);
```

Description

Retrieves multiple files from a remote machine.

The names of the local files are generated algorithmically from the names of the remote files, using the Ftp_Name_Map.Remote_To_Local function. The Local_Roof, Remote_Roof, and Remote_Type parameters are used in this conversion.

Parameters

`From_Remote_File_Set : String := "";`

Specifies a file template for the files to be retrieved. The wildcards used must be those of the remote machine.

`Local_Roof : String := "$";`

Specifies the local parent of the files to be retrieved.

`Remote_Roof : String := Ftp.Current_Remote_Roof;`

Specifies the parent for the files at the remote machine. If the null string is given, retrieved filenames are flattened; that is, the simple names of the source files are mapped to the simple names of the target files.

`Remote_Type : Ftp_Name_Map.Machine_Type := Ftp.Current_Remote_Type;`

Specifies the machine type of the remote machine. This information is used in generating the local filenames, because some differences may exist in directory-naming and filenaming conventions. The default is Rational.

```
Append_To_File : Boolean := False;
```

Specifies, if true, that retrieved files should be appended to local files if they already exist. If the local files do not exist, they are created. If false, the local files are overwritten if they already exist and are created if they do not exist.

```
Response : Profile.Response_Profile := Profile.Get;
```

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

```
Account : String := Ftp_Profile.Account;
```

Specifies the account information supplied to the remote machine when it requires account information. The default value is obtained from the Account switch in the user's session switch file.

Restrictions

The user first must have connected and logged into a remote server before issuing this command. If the session does not have an active FTP connection, an error message is produced and the command has no effect.

Errors

The list of files to be retrieved is generated by requesting the remote server to produce a list of all files matching the specified template. If the remote server does not support this function, the Retrieve_Set procedure does not work.

If a file fails to transfer, the command may attempt to transfer the remaining files or terminate, depending on the specified Response_Profile (Profile.Reaction (Response)).

References

procedure Get_Set

procedure Retrieve_List

function Ftp_Name_Map.Remote_To_Local

procedure Send_Port

```
procedure Send_Port
     (Enabled  : Boolean                  := Ftp_Profile.Send_Port_Enabled;
      Response : Profile.Response_Profile  := Profile.Get);
```

Description

Enables or disables the sending of port information to the remote server.

To ensure that both the local and the remote server are consistent, the FTP protocol supports an option to send the identity of the data connection it will use during a file transfer. This command allows the user to specify whether this option should be active. If Send_Port is enabled, the host identifier and socket number of the local data connection is sent to the remote server before each transfer. The parameter should be set to true for transfer of multiple files.

Parameters

```
Enabled :  Boolean := Ftp_Profile.Send_Port_Enabled;
```

Enables the Send_Port option. If true (the default), the identity of the data connection is sent to the remote server before each file transfer.

```
Response :  Profile.Response_Profile  := Profile.Get;
```

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

renamed procedure Show_Profile

```
procedure Show_Profile (Response : Profile.Response_Profile  := Profile.Get)
                                                      renames Ftp_Profile.Show;
```

Description

Displays the various package Ftp parameters in the session switch file.

Parameters

```
Response :  Profile.Response_Profile  := Profile.Get;
```

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

References

package Ftp_Profile

procedure Status

```
procedure Status (Argument : String                   := "";
                  Response : Profile.Response_Profile := Profile.Get);
```

Description

Displays local status such as the condition of the connection and whether any command is active.

Parameters

`Argument : String := "";`

Specifies an optional argument. Currently this argument is not used.

`Response : Profile.Response_Profile := Profile.Get;`

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

procedure Status_All

```
procedure Status_All (Argument : String                   := "";
                      Response : Profile.Response_Profile := Profile.Get);
```

Description

Displays information for all package Ftp connections originating from this machine.

Parameters

`Argument : String := "";`

Specifies an optional argument. Currently this argument is not used.

`Response : Profile.Response_Profile := Profile.Get;`

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

procedure Store

```
procedure Store
       (From_Local_File : String                     := "<IMAGE>";
        To_Remote_File  : String                     := "";
        Append_To_File  : Boolean                    := False;
        Response        : Profile.Response_Profile  := Profile.Get;
        Account         : String                     := Ftp_Profile.Account);
```

Description

Stores a copy of the local file onto the remote machine to which the session is currently connected.

The transfer is conducted in accordance with all other separately specified parameters (Text, Image, Block, and so on). The user session must already be connected and logged into the remote machine.

Parameters

`From_Local_File : String := "<IMAGE>";`

Specifies the local file to be copied onto the remote machine. The default, "<IMAGE>", references the object associated with the highlighted area in which the cursor is located. If the cursor is not located in the highlighted area, "<IMAGE>" references the object associated with the window in which the cursor is located.

`To_Remote_File : String := "";`

Specifies the destination name for the file on the remote machine. If the null string is specified, the remote filename is derived from the local filename, using the Ftp_Name_Map.Local_To_Remote function.

`Append_To_File : Boolean := False;`

Specifies, if true, that the file should be appended to the remote file if it already exists. If the remote file does not exist, it is created. If false, the remote file is overwritten if it exists and is created if it does not exist.

`Response : Profile.Response_Profile := Profile.Get;`

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

```
Account : String := Ftp_Profile.Account;
```

Specifies the account information supplied to the remote machine when it requires account information. The default value is obtained from the Account switch in the user's session switch file.

Restrictions

The user first must have connected and logged into a remote server before issuing this command. If the session does not have an active FTP connection, an error message is produced and the command has no effect.

References

procedure Retrieve

function Ftp_Name_Map.Local_To_Remote

---

procedure Store_Set

```
procedure Store_Set
(From_Local_File_Set  : String                         := "<IMAGE>";
 Local_Roof           : String                         := "$";
 Remote_Roof          : String                         :=
                                                          Ftp.Current_Remote_Roof;
 Remote_Type          : Ftp_Name_Map.Machine_Type      :=
                                                          Ftp.Current_Remote_Type;
 Append_To_File       : Boolean                        := False;
 Response             : Profile.Response_Profile       := Profile.Get;
 Account              : String                         := Ftp_Profile.Account);
```

Description

Stores multiple files from the local machine onto the currently connected remote machine.

The transfer is conducted in accordance with all other separately specified parameters (Text, Image, Block, and so on).

Parameters

`From_Local_File_Set : String := "<IMAGE>";`

Specifies a file template for the local file to be stored. The default, "<IMAGE>", references the object associated with the highlighted area in which the cursor is located. If the cursor is not located in the highlighted area, "<IMAGE>" references the object associated with the window in which the cursor is located.

`Local_Roof : String := "$";`

Specifies the local ancestor of the files to be stored on the remote machine. If the null string is given, retrieved filenames are flattened; that is, the simple names of the source files are mapped to the simple names of the target files.

`Remote_Roof : String := Ftp.Current_Remote_Roof;`

Specifies the ancestor for the files at the remote machine.

`Remote_Type : Ftp_Name_Map.Machine_Type := Ftp.Current_Remote_Type;`

Specifies the machine type of the remote machine. This information is used in generating the remote filenames, because some differences may exist in directory-naming and filenaming conventions. The default is Rational.

```
Append_To_File : Boolean := False;
```

Specifies, if true, that the files should be appended to the remote files if they already exist. If the remote files do not exist, they are created. If false, the files are overwritten if they exist and are created if they do not exist.

```
Response : Profile.Response_Profile := Profile.Get;
```

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

```
Account : String := Ftp_Profile.Account;
```

Specifies the information account supplied to the remote machine when it requires account information. The default value is obtained from the Account switch in the user's session switch file.

Restrictions

The user first must have connected and logged into a remote server before issuing this command. If the session does not have an active FTP connection, an error message is produced and the command has no effect.

Errors

If a file fails to transfer, the command may attempt to transfer the remaining files or terminate, depending on the specified Response_Profile (Profile.Reaction (Response)).

References

procedure Retrieve

procedure Retrieve_Set

procedure Store

package Ftp_Profile

---

constant Stream

```
Stream : constant Ftp_Defs.Mode_Code := Ftp_Defs.Stream;
```

Description

Defines a constant that imports the Mode_Code value of Stream from Ftp_Defs.

This constant is used to indicate that the data are transmitted as a stream of bytes.

constant Unix

```
Unix : constant Ftp_Name_Map.Machine_Type  := Ftp_Name_Map.Unix;
```

Description

Defines a constant that imports the Machine_Type value of Unix from Ftp_Name-_Map.

This constant is used to denote a machine running under the UNIX operating system.

procedure Use_Account

```
procedure Use_Account
               (Account   : String                   := Ftp_Profile.Account;
                Response  : Profile.Response_Profile  := Profile.Get);
```

Description

Sends a request for the account setting to be changed to the specified value.

Parameters

`Account :  String := Ftp_Profile.Account;`

Specifies the account information supplied to the remote machine when it requires account information. The default value is obtained from the Account switch in the user's session switch file.

`Response :  Profile.Response_Profile  := Profile.Get;`

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

procedure Use_Mode

```
procedure Use_Mode
         (Value    : Ftp_Defs.Mode_Code         := Ftp_Profile.Transfer_Mode;
          Response : Profile.Response_Profile := Profile.Get;
          Account  : String                     := Ftp_Profile.Account);
```

Description

Sends a request for the transfer mode setting to be changed to the specified value.

The client first determines whether it supports the transfer mode; if it does, the request is sent to the remote server. If a positive response is received, the local setting is updated.

Parameters

```
Value :  Ftp_Defs.Mode_Code  := Ftp_Profile.Transfer_Mode;
```

Specifies the new transfer mode.

```
Response :  Profile.Response_Profile  := Profile.Get;
```

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

```
Account :  String := Ftp_Profile.Account;
```

Specifies the account information supplied to the remote machine when it requires account information. The default value is obtained from the Account switch in the user's session switch file.

Restrictions

The user first must have connected and logged into a remote server before issuing this command. If the session does not have an active FTP connection, an error message is produced and the command has no effect.

procedure Use_Remote_Roof

```
procedure Use_Remote_Roof
              (Value    : String                    := Ftp_Profile.Remote_Roof;
               Response : Profile.Response_Profile  := Profile.Get);
```

Description

Specifies the roof value to be used for the remote machine.

Parameters

```
Value :  String := Ftp_Profile.Remote_Roof;
```

Specifies a value for the remote roof.

```
Response :  Profile.Response_Profile  := Profile.Get;
```

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

References

procedure Retrieve_List

procedure Retrieve_Set

procedure Store_Set

package Ftp_Name_Map

procedure Use_Remote_Type

```
procedure Use_Remote_Type
          (Value    : Ftp_Name_Map.Machine_Type  := Ftp_Profile.Remote_Type;
           Response : Profile.Response_Profile   := Profile.Get);
```

Description

Specifies the machine type for the current connection.

This information is used when resolving differences in the directory-naming conventions. When a connection is formed to a remote machine, the service has no knowledge of the type of machine to which it is connected. Therefore, the type must be specified either through this command or as a parameter to the transfer commands.

Parameters

```
Value :  Ftp_Name_Map.Machine_Type  := Ftp_Profile.Remote_Type;
```

Specifies the type of machine for the current connection.

```
Response :  Profile.Response_Profile  := Profile.Get;
```

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

References

procedure Retrieve_List

procedure Retrieve_Set

procedure Store_Set

procedure Use_Structure

```
procedure Use_Structure
    (Value     : Ftp_Defs.Structure_Code   := Ftp_Profile.Transfer_Structure;
     Response  : Profile.Response_Profile  := Profile.Get;
     Account   : String                    := Ftp_Profile.Account);
```

Description

Sends a request for the transfer structure setting to be changed to the specified value.

The client first determines whether it supports the structure; if it does, the request is sent to the remote server. If a positive response is received, the local setting is updated.

Parameters

`Value : Ftp_Defs.Structure_Code := Ftp_Profile.Transfer_Structure;`

Specifies the new transfer structure.

`Response : Profile.Response_Profile := Profile.Get;`

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

`Account : String := Ftp_Profile.Account;`

Specifies the account information supplied to the remote machine when it requires account information. The default value is obtained from the Account switch in the user's session switch file.

Restrictions

The user first must have connected and logged into a remote server before issuing this command. If the session does not have an active FTP connection, an error message is produced and the command has no effect.

procedure Use_Type

```
procedure Use_Type
        (Value    : Ftp_Defs.Type_Code          := Ftp_Profile.Transfer_Type;
         Response : Profile.Response_Profile := Profile.Get;
         Account  : String                   := Ftp_Profile.Account);
```

Description

Sends a request for the transfer type setting to be changed to the specified value.

The client first determines whether it supports the structure; if it does, the request is sent to the remote server. If a positive response is received, the local setting is updated.

Parameters

`Value : Ftp_Defs.Type_Code := Ftp_Profile.Transfer_Type;`

Specifies the new transfer type setting.

`Response : Profile.Response_Profile := Profile.Get;`

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

`Account : String := Ftp_Profile.Account;`

Specifies the account information supplied to the remote machine when it requires account information. The default value is obtained from the Account switch in the user's session switch file.

Restrictions

The user first must have connected and logged into a remote server before issuing this command. If the session does not have an active FTP connection, an error message is produced and the command has no effect.

constant Vax

```
Vax : constant Ftp_Name_Map.Machine_Type  := Vms;
```

Description

Defines a constant that imports a Machine_Type value of Vms from Ftp_Name_Map.

This constant is used to denote a DEC machine using the VMS operating system.

constant Vms

```
Vms : constant Ftp_Name_Map.Machine_Type := Ftp_Name_Map.Vms;
```

Description

Defines a constant that imports a Machine_Type value of Vms from Ftp_Name-_Map.

This constant is used to denote a DEC machine using the VMS operating system.

end Ftp;

RATIONAL

package Ftp_Defs

Package Ftp_Defs provides types that are used by packages Ftp and File_Transfer and other networking software.

constant Default_Mode

```
Default_Mode : Mode_Code := Stream;
```

Description

Specifies the default transfer mode used by the FTP protocol.

constant Default_Structure

```
Default_Structure : Structure_Code := File;
```

Description

Specifies the default transfer structure used by the FTP protocol.

constant Default_Type

```
Default_Type : Type_Code := Ascii;
```

Description

Specifies the default transfer type used by the FTP protocol.

type Dio_Pointer

```
type Dio_Pointer is access Device_Independent_Io.File_Type;
```

Description

Defines access to the type of file used during package Ftp transfer operations.

This command allows the user to share access to the transfer file with the package Ftp programmatic interface. In carrying out nonbinary transfers, the local file transfer server uses a specific file handle when reading from or writing into a local file. These operations are done using package !Io.Device_Independent_Io. In some cases, a user may want to share access to this file handle. For example, a user may want to have a program generate data into a local file and then transfer that file across the network. One way would be to write the data into a named file and then request file transfer to transfer the file given its name. Another method would be to open this local file using the file handle from package File_Transfer, write the information into the file, and then reset the file for reading. File transfer is then requested to transfer the file.

Example

```
declare
  The_File : Ftp_Defs.Dio_Pointer;
begin
  The_File := File_Transfer.Dio_File_Of (Connection);
  Device_Independent_Io.Open (File => The_File.All,
                              Mode => Device_Independent_Io.Out_File);
              --- Open local temp file.
   .
   .
   .
  Device_Independent_Io.Write (File => The_File.All,
                               "some data");
   .          --- Write information to the file.
   .
   .

  Device_Independent_Io.Reset
        (The_File.All,Device_Independent_Io.In_File);
              --- Reset the file so the file transfer server can
              --- read from it.

  File_Transfer.Start_Store (Connection => Connection,
                             Remote_Pathname => "remotefile");
              --- Start the transfer with the form specifying no
              --- local filename.  This form will use the open file
              --- handle.
```

```
     File_Transfer.Command_Status  (Connection => Connection,
                                     Status => Status);
                    --- Check on the status of the transfer.
                    --- This procedure returns when the transfer
                    --- is complete.

     Device_Independent_Io.Close  (File => The_File.All);
                    --- Since the user had control of the file, it
                    --- will need to be closed at the end of the transfer.
   end;
```

References

package File_Transfer

type Ftp_Commands

```
type Ftp_Commands is (Login, Set_User, Set_Pass, Set_Account, Set_Type,
                      Set_Stru, Set_Mode, Set_Allocation, Send_File,
                      Send_File_Append, Retr_File, List_Directory,
                      Nlst_Directory, Do_Cwd, Do_Delete, Do_Help, Do_Stat,
                      Do_Port, Do_Pasv, Do_Site, Verbatim, Do_Quit, Noop);
```

Description

Specifies the various operations performed by the FTP service.

Of interest when working with the programmatic interface, these operations give the user the ability to check the current command status.

Enumerations

`Do_Cwd`

Changes remote current working directory.

`Do_Delete`

Deletes file at remote host.

`Do_Help`

Gets help from remote.

`Do_Pasv`

Requests passive data operation.

`Do_Port`

Sends data port information.

`Do_Quit`

Logs off.

`Do_Site`

Sends FTP protocol "SITE" command.

`Do_Stat`

Gets status from remote.

`List_Directory`
Gets verbose directory listing.

`Login`
Forms the initial connection.

`Nlst_Directory`
Gets name list of directory.

`Noop`
Has no effect.

`Retr_File`
Retrieves file from remote.

`Send_File`
Sends file to remote.

`Send_File_Append`
Sends file and appends to remote file.

`Set_Account`
Sets account information.

`Set_Allocation`
Sets space allocation.

`Set_Mode`
Sets transfer mode information.

`Set_Pass`
Sets password information.

`Set_Stru`
Sets transfer structure information.

`Set_Type`
Sets transfer type information.

`Set_User`
Sets user information.

`Verbatim`
Sends an arbitrary FTP protocol command.

References

function File_Transfer.Most_Recent_Command

---

function Ftp_Product_Is_Installed

```
function Ftp_Product_Is_Installed return Boolean;
```

Description

Determines whether the FTP utility is installed on the user's system.

Parameters

```
return Boolean;
```

Returns a Boolean value.

type Mode_Code

```
type Mode_Code is (Stream, Block, Compressed);
```

Description

Defines the transmission modes for the transfer of data over the FTP data connection.

Enumerations

`Block`

Transmits a file as a series of data blocks preceded by one or more header bytes. The header bytes contain a count field and a descriptor code. The count field indicates the total length of the data block in bytes. The descriptor code indicates structure items such as End_Of_File and End_Of_Record.

`Compressed`

Compresses and sends replications of bytes and filler as control information describing the number of repetitions of a given byte or filler.

`Stream`

Transmits data as a stream of bytes. There is no restriction on the representation type used.

Restrictions

Stream mode is the only transfer mode supported by the Rational FTP implementation.

References

package File_Transfer

package Ftp

type Pio_Handle

```
type Pio_Handle is
    record
        File_Handle : Polymorphic_Io.Handle;
        Position    : Polymorphic_Io.File_Position;
    end record;
```

Description

Defines the components of the file handle used when dealing with binary transfers of data.

This command is used in the transfer of nonbyte-aligned data.

References

package File_Transfer

type Pio_Pointer

```
type Pio_Pointer is access Pio_Handle;
```

Description

Defines access to the type of file used during a binary transfer operation.

This type allows the user to share access to the transfer file with the package Ftp programmatic interface.

Example

At this point, there is a connection set up for binary transfers:

```
declare
  The_File : Ftp_Defs.Pio_Pointer;
begin
  The_File := File_Transfer.Pio_File_Of (Connection);
  Polymorphic_Io.Open (The_Handle => The_File.Handle,
                       Mode => Polymorphic_Io.Read_Only,
                       File_Name => "Movefile",
                       Status => Poly_Status);
  File_Transfer.Start_Store (Connection => Connection,
                             Remote_Pathname => "Remotefile");
  File_Transfer.Command_Status (Connection => Connection,
                                Status => Status);
  Polymorphic_Io.Close (File => The_File.Handle,
                        Status => Poly_Status);
end;
```

References

package File_Transfer

type Status_Code

```
type Status_Code is (Command_In_Progress, Successful, Need_Password,
                     Need_Account, Not_Used, Transfer_Started,
                     Transfer_Complete, Transfer_Failed, Not_Logged_In,
                     Syntax_Error, Bad_Sequence, No_Local_Support,
                     Command_Not_Implemented, Param_Not_Implemented,
                     Local_Pasv_Error, Remote_Directory_Error, Timed_Out,
                     Network_Error, Invalid_Use, Unknown_Error);
```

Description

Specifies the status of the current FTP operation.

Enumerations

`Bad_Sequence`

Indicates that sequenced commands (that is, username followed by password) were done out of sequence.

`Command_In_Progress`

Indicates that the command is active pending a response.

`Command_Not_Implemented`

Indicates that the remote server does not implement the requested command.

`Invalid_Use`

Indicates that the connection is not open or the caller is not a valid user of the connection.

`Local_Pasv_Error`

Indicates that a Send_Pasv request got a successful response from the remote server but the local server could not interpret the returned port information.

`Need_Account`

Indicates that account information is needed to proceed.

`Need_Password`

Indicates that the command has completed successfully and the password is needed next.

Network_Error

Indicates that some problem has occurred on the network. The connection may have been lost.

No_Local_Support

Indicates that the local transfer worker has no local support for the requested option.

Not_Logged_In

Indicates that the command failed because the user login procedure failed or the user never logged in.

Not_Used

Indicates that the command is not needed at the remote server (that is, no account is needed).

Param_Not_Implemented

Indicates that the remote server does not support the specified option for the command (that is, the page structure transfer may not be supported).

Remote_Directory_Error

Indicates a file or directory access error.

Successful

Indicates that the command has completed successfully.

Syntax_Error

Indicates a command syntax error (that is, the line is too long).

Timed_Out

Indicates that no response was received by the local server from the last request sent.

Transfer_Complete

Indicates that the transfer has completed successfully.

Transfer_Failed

Indicates that the transfer failed. Refer to Transfer_Status type to determine the reason for the failure.

`Transfer_Started`

Indicates that the transfer of data is currently being performed.

`Unknown_Error`

Indicates that an error response that could not be classified was received.

References

function File_Transfer.Most_Recent_Command_Status

---

type Structure_Code

```
type Structure_Code is  (File, Recrd, Page);
```

Description

Defines the various representations for the structure of the file during the transfer.

Enumerations

`File`
Indicates that the file is a continuous sequence of data bytes.

`Page`
Indicates that the file is made up of independent, indexed pages.

`Recrd`
Indicates that the file is made up of sequential records.

Restrictions

File structure is the only structure supported by the Rational FTP implementation.

References

package File_Transfer

package Ftp

type Transfer_Status

```
type Transfer_Status is (In_Progress, Ok, Local_Error, File_Error,
                          Line_Error, Open_Failed, Transfer_Abort,
                          Remote_File_Unavailable, Remote_File_Error,
                          Storage_Error, Unknown_Page_Type, Filename_Bad,
                          Comm_Line_Error, Unknown_Error);
```

Description

Defines the various status codes associated with data transfers.

These status values are meaningful only after a data transfer has been performed.

Enumerations

`Comm_Line_Error`

Indicates that the command link was lost during the data transfer, which means that the outcome of the data transfer could not be determined.

`File_Error`

Indicates that a transfer failed because of a local file error.

`Filename_Bad`

Indicates that the filename specified for the remote machine was not allowed.

`In_Progress`

Indicates that a transfer is currently active.

`Line_Error`

Indicates a problem with the data line; the transfer could not be completed.

`Local_Error`

Indicates that the transfer failed because of a local error in processing.

`Ok`

Indicates that a transfer has successfully completed.

`Open_Failed`

Indicates that a data connection could not be opened to start the transfer.

`Remote_File_Error`

Indicates that a file error occurred at the remote machine.

`Remote_File_Unavailable`

Indicates that the file was unavailable at the remote site.

`Storage_Error`

Indicates that the transfer failed because of storage limitations at the remote machine.

`Transfer_Abort`

Indicates that the file transfer has been aborted.

`Unknown_Error`

Indicates that an error response that could not be classified was received for the last transfer.

`Unknown_Page_Type`

Indicates that the remote machine did not recognize the page structure used for data transfer.

References

function File_Transfer.Most_Recent_Transfer_Status

---

type Type_Code

```
type Type_Code is (Ascii, Ebcdic, Image, Binary, Local_Binary, Local_Byte,
                   Ascii_Cc, Ebcdic_Cc, Ascii_Telnet, Ebcdic_Telnet);
```

Description

Defines the data representation for information transfer over the data connection during FTP file transfers.

Enumerations

`Ascii`

Transfers text files. This is the default type for data transfer and is intended for the transfer of text files. The data are transmitted using the standard eight-bit NVT-ASCII representation for the characters.

`Ascii_Cc`

Transfers ASCII type with ASA (FORTRAN) vertical format control. Currently not supported by the Rational implementation.

`Ascii_Telnet`

Transfers ASCII type with Telnet vertical format control. Currently not supported by the Rational implementation.

`Binary`

Transmits binary data that are not byte-aligned. This type is similar to Image, but it is intended for true binary transfer of data. This type is supported only by Rational FTP protocol implementation. Binary data are sent with the final bits padded to a byte boundary. An additional final byte specifies the number of bits of padding added. The receiving server uses this information to remove the padded bits and to store only the bits from the original file.

`Ebcdic`

Transmits text files using eight-bit EBCDIC character representation.

`Ebcdic_Cc`

Transfers EBCDIC type with ASA (FORTRAN) vertical format control. Currently not supported by the Rational implementation.

`Ebcdic_Telnet`

Transfers EBCDIC type with Telnet vertical format control. Currently not supported by the Rational implementation.

`Image`

Transmits binary data as contiguous bits of eight-bit transfer bytes. The receiving site stores the data as contiguous bytes. It should be noted that the transfer is in bytes.

`Local_Binary`

Transmits binary data that are not byte-aligned to or from a server that does not support the Binary mode. Data are sent with the final byte padded and the padding information added. The receiving server does not remove them but instead stores all the bytes sent. If a file with this padding information is retrieved using Local_Binary, the padding information is removed and only the binary image is stored in the local file. This mode allows storing of binary files on non-Rational systems to be processed there or later retrieved.

`Local_Byte`

Transfers data in logical bytes of the size specified with the type. Currently not supported by the Rational implementation.

Restrictions

Currently, Ascii_Cc, Ascii_Telnet, Ebcdic_Cc, Ebcdic_Telnet, and Local_Byte are not supported by the Rational implementation.

References

package File_Transfer

package Ftp

end Ftp_Defs;

RATIONAL

package Ftp_Name_Map

Package Ftp_Name_Map defines mappings between Rational object names and filenames on other machines. When package Ftp is used to transfer a set of files between machines, the package uses these mappings to convert filenames. The mappings are intended to preserve the information in filenames and to produce legal names on the target machine. Directory names are mapped to directory names, and filenames are mapped to filenames. Punctuation characters are mapped to legal values.

The mapping functions are built around the idea of a *roof*. Basically, a roof is an ancestor directory of the file being transferred; that is, it contains the file or it contains a directory that contains the file, and so on. For each set of transferred files, the user specifies a Local_Roof (an ancestor directory on the local machine) and a Remote_Roof (an ancestor directory on the remote machine). Filenames are mapped on the assumption that the directory subtrees under the two roofs are isomorphic; that is, each directory under the local roof has a correspondingly named directory under the remote roof. This is intended to help the user maintain copies of the same objects on two machines.

If the source file is immediately within the directory named by the source roof, the target filename is immediately within the directory named by the target roof. If there are directory levels between the source roof and the source filename, the corresponding directory levels are interposed between the target roof and the target filename. If a source name is outside the source roof (that is, the source roof does not identify a parent directory of the source file), the target filename is immediately under the target roof, with a simple name derived from the simple filename of the source file.

To flatten a file set (that is, to transfer files from many directories on the source machine into a single directory on the target machine), the user specifies a null source roof (""). This causes the mapping functions to map all filenames into the directory identified by the target roof, with simple names mapped from their source simple names.

The mappings are not invertible; mapping to a remote filename and then mapping back does not always result in the same filename.

The mappings differ for various types of remote machines. The differences primarily concern punctuation and restrictions on names.

For Rational objects other than files, the mapping functions try to put information about the type of the file in the filename. For example, an Ada unit named My_Program will be mapped to the filename My_Program.Ada or My_Program-_Ada. This is useful because Ada units are transferred as text. When a transferred unit arrives at its destination, it will be Ada source, even though it may have been a compiled program originally.

function Local_To_Remote

```
function Local_To_Remote (File_Name   : String;
                          Local_Roof  : String;
                          Remote_Roof : String;
                          Remote_Type : Machine_Type) return String;
```

Description

Maps the name of a local object to its remote form.

This function converts the filename on the local machine to a legal filename on the target host. Directory names are converted to legal directory names on the target host. Punctuation characters in the host filenames and directories are converted to legal values on the target host.

The function attempts to resolve the file and directory names in their context. If it cannot resolve the names, the function returns the best resolution possible.

This function can be used to ascertain the mappings before the file is sent to the target host.

Parameters

`File_Name : String;`

Specifies the name of an object on the local machine.

`Local_Roof : String;`

Specifies the local name of a directory that is an ancestor of the local object.

`Remote_Roof : String;`

Specifies the remote name of a directory that is an ancestor of the remote object.

`Remote_Type : Machine_Type;`

Specifies the type of the remote machine.

`return String;`

Returns the remote name of a file on the remote machine. This name always begins with the Remote_Roof.

Errors

If the Local_Roof is not an ancestor of the local object, the name returned is immediately under the Remote_Roof.

Example

```
text_io.put (ftp_name_map.local_to_remote
                    ("phone'body",
                     "!users.wjh.phoneproblem",
                     "[users.wjh]",
                     "vms"));

[USERS.WJH]PHONE_BODY


text_io.put (ftp_name_map.local_to_remote
                    ("phone#$^&().ada",
                     "!users.wjh",
                     "[users.wjh]",
                     "vms"));

[USERS.WJH]PHONE#$^&_._.ADA
```

type Machine_Type

```
type Machine_Type is new String;

Rational : constant Machine_Type := "Rational";
Unix     : constant Machine_Type := "Unix";
Aos      : constant Machine_Type := "Aos";
Vms      : constant Machine_Type := "Vms";
Mvs      : constant Machine_Type := "Mvs";
```

Description

Specifies the types of remote machines supported by the mapping functions.

Name mappings differ for various types of remote machines. The differences primarily concern punctuation and restrictions on names.

"Rational" denotes a machine running the Rational Environment. Rational pathnames use the exclamation mark (!) to denote the root and the period (.) to separate the simple names of directories. Simple names are Ada simple names.

"Unix" denotes a machine running the UNIX operating system. UNIX pathnames use the slash (/) to denote the root and to separate the simple names of directories. The period (.) is commonly used within a simple name to separate the object name and object class.

"Aos" denotes a machine running the Data General AOS operating system. AOS pathnames use the colon (:) to denote the root and to separate the simple names of directories. The period (.) is commonly used within a simple name to separate the object name and object class.

"Vms" denotes a machine running the DEC VMS operating system. VMS pathnames use brackets ([]) to enclose directory pathnames. Within the brackets, a period (.) separates the simple names of directories. The period is also used within a simple filename to separate the object name and object class.

"Mvs" denotes a machine running the IBM MVS operating system. "Mvs" is not currently supported.

function Remote_To_Local

```
function Remote_To_Local (File_Name   : String;
                          Local_Roof  : String;
                          Remote_Roof : String;
                          Remote_Type : Machine_Type) return String;
```

Description

Maps the name of a remote file to its local form.

This function converts the filename on the remote machine to a legal filename on the local host. Directory names are converted to legal directory names on the local host. Punctuation characters in the host filenames and directories are converted to legal values on the local host.

The function attempts to resolve the file and directory names in their context. If it cannot resolve the names, the function returns the best resolution possible.

This function can be used to ascertain the mappings before the file is sent to the local host.

Parameters

`File_Name : String;`

Specifies the remote name of a file on the remote machine.

`Local_Roof : String;`

Specifies the local name of a directory that is an ancestor of the local object.

`Remote_Roof : String;`

Specifies the remote name of a directory that is an ancestor of the remote object.

`Remote_Type : Machine_Type;`

Specifies the type of the remote machine.

`return String;`

Returns the name of a file on the local machine. This name always begins with the Local_Roof.

Errors

If the Remote_Roof is not an ancestor of the remote object, the name returned is immediately under the Local_Roof.

Example

```
    text_io.put (ftp_name_map.remote_to_local
                        ("phone_body",
                         "!users.wjh.phoneproblem",
                         "[users.wjh]",
                         "vms"));

    !USERS.WJH.PHONEPROBLEM.PHONE_BODY


    text_io.put (ftp_name_map.remote_to_local
                        ("phone_body#$^&_._.ada",
                         "!users.wjh",
                         "[users.wjh]",
                         "vms"));

    !USERS.WJH.PHONEPROBLEM.PHONE_BODY#_^ &_V_V_V_ADA
```

end Ftp_Name_Map;

RATIONAL

package Ftp_Product

Package Ftp_Product provides a way to check whether the FTP product has been installed on your machine.

function Is_Installed

```
function Is_Installed return Boolean;
```

Description

Returns true if the FTP product has been installed on your machine.

exception Is_Not_Installed

```
Is_Not_Installed : exception;
```

Description

Defines an exception that is raised when an FTP subprogram is called if the FTP product is not installed on your machine.

end Ftp_Product;

RATIONAL

package Ftp_Profile

The main purpose of this package is to provide default values for the FTP user interface (package Ftp).

Many parameter values are virtually constant for a given user. For common applications, it is much easier to have the values of these parameters automatically set to the commonly used values rather than to specify parameter values each time the command is used. The Rational Environment supplies two types of switches to do this:

- The *session switches* set parameter values at login. FTP uses these session values for the duration of the login session.
- The *library switches* set parameter values for a given library. For transfers done from a Command window inside a given library, FTP uses the switch values associated with that library.

Switches are stored in files. The user can edit the switch values in these files by means of the switch file object editor. The !Commands.Switches.Edit command edits library switch files. The !Commands.Switches.Edit_Session_Attributes command edits session switch files. Refer to the Session and Job Management (SJM) and Library Management (LM) books of the *Rational Environment Reference Manual* for further information on switches.

There is a switch to provide a value for most FTP parameters. If the user does not give the file transfer command a parameter value in the FTP Command window:

1. FTP tries to get a value from the appropriate library switch file.
2. If no value has been explicitly set for the library switch, FTP tries to take the parameter value from the user's session switch file.
3. If no value has been set for the session switch, FTP tries to take the parameter value from one of the following files:
   - Remote password file for username or password
   - Remote session file for account name
4. If no value has been set in the remote files, FTP takes the standard default.

By keeping this search sequence in mind, the user can set a combination of library, session, and standard values for FTP's parameter values. These values are available through package Ftp_Profile.

The Rational Environment supports the following switches for FTP:

Account	Auto_Login
Password	Prompt_For_Account
Prompt_For_Password	Remote_Directory
Remote_Machine	Remote_Roof
Remote_Type	Send_Port_Enabled
Transfer_Mode	Transfer_Structure
Transfer_Type	Username

function Account

```
function Account return String;
```

Description

Specifies the account information from the current switch files.

This information is supplied to the remote host when it requires account information for login or file access. The default value is the null string.

Parameters

```
return String;
```

Returns the account value for the current session. The default is the null string ("").

References

procedure Ftp.Connect

procedure Ftp.Login

function Auto_Login

```
function Auto_Login return Boolean;
```

Description

Specifies the current setting from the current switch files, indicating whether automatic login should occur when the user connects to a remote machine.

The default value is false.

Parameters

```
return Boolean;
```

Returns the Boolean value indicating the automatic login setting. If true, the Ftp.Connect procedure attempts to log the user session into the remote FTP server automatically. The default is false.

References

procedure Ftp.Connect

function Current_Remote_Roof

```
function Current_Remote_Roof return String;
```

Description

Returns the current roof value for the remote host.

This value is used when moving multiple files to or from the remote host to indicate the outermost point in the directory structure.

Parameters

```
return String;
```

Returns the roof value as a string.

References

procedure Ftp.Retrieve_Set

procedure Ftp.Store_Set

procedure Ftp.Use_Remote_Roof

package Ftp_Name_Map

function Current_Remote_Type

```
function Current_Remote_Type return Ftp_Name_Map.Machine_Type;
```

Description

Returns a value indicating the type of machine currently specified for the remote host.

This information is used during automatic generation of remote filenames to determine any translations needed for pathname characters marking directory structure information.

Parameters

```
return Ftp_Name_Map.Machine_Type;
```

Returns the machine type.

References

procedure Ftp.Retrieve_Set

procedure Ftp.Store_Set

procedure Ftp.Use_Remote_Type

package Ftp_Name_Map

function Password

```
function Password return String;
```

Description

Specifies the password information from the current switch files.

This information is supplied to the remote host when required for login. The default value is the null string.

Parameters

```
return String;
```

Returns the string value containing the password. The default is the null string ("").

References

procedure Ftp.Connect

procedure Ftp.Login

function Prompt_For_Account

```
function Prompt_For_Account return Boolean;
```

Description

Prompts for account information.

If the user has not already supplied account information and if Prompt_For_Account is set to true, FTP will request that an account be supplied. FTP will not echo the account string.

Parameters

```
return Boolean;
```

Returns the current function setting.

References

procedure Ftp.Connect

procedure Ftp.Login

function Prompt_For_Password

```
function Prompt_For_Password return Boolean;
```

Description

Prompts for a password.

If the user has not already supplied a password and if Prompt_For_Password is set to true, FTP will request that a password be supplied. FTP will not echo the password.

Parameters

```
return Boolean;
```

Returns the current function setting.

References

procedure Ftp.Connect

procedure Ftp.Login

function Remote_Directory

```
function Remote_Directory return String;
```

Description

Specifies the directory pathname from the current switch files to be used when setting the context on the remote machine.

The default value is the null string.

Parameters

```
return String;
```

Returns the string value containing the pathname for the remote context. The default is the null string ("").

References

procedure Ftp.Change_Working_Directory

procedure Ftp.Connect

procedure Ftp.Cwd

procedure Ftp.Get

procedure Ftp.Put

function Remote_Machine

```
function Remote_Machine return String;
```

Description

Specifies the current machine name from the current switch files, indicating to which machine an FTP connection should occur.

The default value is the null string.

Parameters

```
return String;
```

Returns the string value containing the name of a machine. The default is the null string ("").

References

procedure Ftp.Connect

procedure Ftp.Get

procedure Ftp.Put

function Remote_Roof

```
function Remote_Roof return String;
```

Description

Sets the remote roof directory to be used when transferring to or from the remote machine.

The default value is the null string, which usually indicates the current working directory.

Parameters

```
return String;
```

Returns the string value specifying the roof for the remote machine. The default is the null string ("").

References

procedure Ftp.Get_Set

procedure Ftp.Put_Set

procedure Ftp.Retrieve_Set

procedure Ftp.Store_Set

procedure Ftp.Use_Remote_Roof

function Remote_Type

```
function Remote_Type return Ftp_Name_Map.Machine_Type;
```

Description

Specifies the machine type for the remote machine.

The default value is Ftp_Name_Map.Rational.

Parameters

`return Ftp_Name_Map.Machine_Type;`
Returns the machine type information. Possible values are:

- Rational
- Aos
- Unix
- Vms

The default is Rational.

References

procedure Ftp.Connect

procedure Ftp.Get_Set

procedure Ftp.Put_Set

procedure Ftp.Use_Remote_Type

function Send_Port_Enabled

```
function Send_Port_Enabled return Boolean;
```

Description

Specifies the current setting from the current switch files, indicating whether the data port identifier command should be sent before each transfer.

The default value is true.

Parameters

```
return Boolean;
```

Returns the Boolean value indicating the setting of the switch. The default is true.

References

procedure Ftp.Connect

procedure Ftp.Get

procedure Ftp.Login

procedure Ftp.Put

procedure Ftp.Send_Port

function Transfer_Mode

```
function Transfer_Mode return Ftp_Defs.Mode_Code;
```

Description

Specifies the current setting from the current switch files, indicating the mode of transfer to be used during file transfers.

The default value is Ftp_Defs.Default_Mode.

Parameters

`return Ftp_Defs.Mode_Code;`
Returns the mode value for transfers.

References

procedure Ftp.Connect

procedure Ftp.Get

procedure Ftp.Put

procedure Ftp.Use_Mode

function Transfer_Structure

```
function Transfer_Structure return Ftp_Defs.Structure_Code;
```

Description

Specifies the current setting from the current switch files, indicating the structure of transfer to be used during file transfers.

The default value is Ftp_Defs.Default_Structure.

Parameters

`return Ftp_Defs.Structure_Code;`
Returns the structure value for transfers.

References

procedure Ftp.Connect

procedure Ftp.Get

procedure Ftp.Put

procedure Ftp.Use_Structure

function Transfer_Type

```
function Transfer_Type return Ftp_Defs.Type_Code;
```

Description

Specifies the current setting from the current switch files, indicating the type of transfer to be used during file transfers.

The default value is Ftp_Defs.Default_Type.

Parameters

```
return Ftp_Defs.Type_Code;
```

Returns the type value for transfers.

References

procedure Ftp.Connect

procedure Ftp.Get

procedure Ftp.Put

procedure Ftp.Use_Type

function Username

```
function Username return String;
```

Description

Specifies the current value for the current switch files, indicating the username to be used for login to a remote FTP server.

The default value is the username of the current session.

Parameters

`return String;`
Returns the string value containing the FTP username. The default is the username of the current session.

References

procedure Ftp.Connect

procedure Ftp.Login

end Ftp_Profile;

generic package Transfer_Generic

Package Transfer_Generic, a generic version of package Ftp, is used with remote machines whose naming conventions are not supported by package Ftp_Name_Map. Users can supply their own customized name-mapping functions (as generic parameters) to replace the standard name mappings provided by package Ftp_Name_Map.

```
with Ftp_Name_Map; ...

generic
    with function Local_To_Remote (File_Name : String;
                                    Local_Roof : String;
                                    Remote_Roof : String;
                                    Remote_Type : Ftp_Name_Map.Machine_Type)
                       return String;

    with function Remote_To_Local (File_Name : String;
                                    Local_Roof : String;
                                    Remote_Roof : String;
                                    Remote_Type : Ftp_Name_Map.Machine_Type)
                       return String;

package Transfer_Generic is
    procedure Put (From_Local_File : String := ""; ...);
    procedure Get (From_Remote_File : String := ""; ...);
    procedure Store (From_Local_File : String := ""; ... );
    procedure Retrieve (From_Remote_File : String := ""; ...);
    procedure Put_Set (From_Local_File_Set : String := ""; ... );
    procedure Get_Set (From_Remote_File_Set : String := ""; ... );
    procedure Get_List (Remote_File_List : String := ""; ... );
    procedure Store_Set (From_Local_File_Set : String := ""; ... );
    procedure Retrieve_Set (From_Remote_File_Set : String := ""; ... );
    procedure Retrieve_List (Remote_File_List : String := ""; ... );
end Transfer_Generic;
```

The behavior of the Transfer_Generic procedures is like that of the correspondingly named procedures in package Ftp, except that filenames are mapped using the generic formals. In fact, the Ftp procedures are implemented as calls to an instantiation of Transfer_Generic on the standard Ftp_Name_Map functions.

Ftp commands that do not map filenames do not have corresponding commands in package Transfer_Generic.

procedure Get

```
procedure Get
(From_Remote_File    : String                         := "";
 To_Local_File       : String                         := "";
 Remote_Machine      : String                         :=
                                          Ftp_Profile.Remote_Machine;
 Username            : String                         := Ftp_Profile.Username;
 Password            : String                         := Ftp_Profile.Password;
 Account             : String                         := Ftp_Profile.Account;
 Remote_Directory    : String                         :=
                                        Ftp_Profile.Remote_Directory;
 Remote_Type         : Ftp_Name_Map.Machine_Type      :=
                                             Ftp_Profile.Remote_Type;
 Append_To_File      : Boolean                        := False;
 Transfer_Type       : Ftp_Defs.Type_Code             :=
                                           Ftp_Profile.Transfer_Type;
 Transfer_Mode       : Ftp_Defs.Mode_Code             :=
                                           Ftp_Profile.Transfer_Mode;
 Transfer_Structure  : Ftp_Defs.Structure_Code        :=
                                      Ftp_Profile.Transfer_Structure;
 Send_Port           : Boolean                        :=
                                       Ftp_Profile.Send_Port_Enabled;
 Response            : Profile.Response_Profile       := Profile.Get);
```

Description

Retrieves a file from a remote host.

This procedure is used as a one-step operation to retrieve a file from a remote host. It forms a connection that logs in the user and retrieves the file. At the end of the transfer, the connection is disconnected.

Parameters

`From_Remote_File : String := "";`

Specifies the file to be retrieved from the remote host. A filename must be given for this parameter.

`To_Local_File : String := "";`

Specifies the local file into which the copy is to be written. If the null string is given, then a local name is built using the remote name and the Remote_To_Local generic formal function.

`Remote_Machine : String := Ftp_Profile.Remote_Machine;`

Specifies the remote machine to which the connection should be formed.

```
Username : String := Ftp_Profile.Username;
```

Specifies the username to be supplied to the remote server for login.

```
Password : String := Ftp_Profile.Password;
```

Specifies the password to be supplied to the remote server for login.

```
Account : String := Ftp_Profile.Account;
```

Specifies the account information supplied to the remote machine when it requires account information. The default value is obtained from the Account switch in the user's session switch file.

```
Remote_Directory : String := Ftp_Profile.Remote_Directory;
```

Specifies the remote directory context for this transfer. The value "" means that the Get procedure does not explicitly set the directory.

```
Remote_Type : Ftp_Name_Map.Machine_Type := Ftp_Profile.Remote_Type;
```

Specifies the machine type of the remote machine. This information can be used when forming the local filename. The default is the value provided by the Remote_Type switch in the session switch file.

```
Append_To_File : Boolean := False;
```

Specifies, if true, that the retrieved file should be appended to the local file if it already exists. If the file does not exist, it is created. If false, the local file is overwritten if it exists and is created if it does not exist.

```
Transfer_Type : Ftp_Defs.Type_Code := Ftp_Profile.Transfer_Type;
```

Specifies the representation type to be used while transferring the file.

```
Transfer_Mode : Ftp_Defs.Mode_Code := Ftp_Profile.Transfer_Mode;
```

Specifies the transfer mode to be used while transferring the file.

```
Transfer_Structure : Ftp_Defs.Structure_Code :=
                                   Ftp_Profile.Transfer_Structure;
```

Specifies the transfer structure to be used while transferring the file.

```
Send_Port : Boolean := Ftp_Profile.Send_Port_Enabled;
```

Specifies whether the server should be explicitly informed of the port to use for data transfer. It is recommended that this parameter be set to true (the default) when transferring multiple files in a single session.

```
Response : Profile.Response_Profile := Profile.Get;
```

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

Restrictions

If a connection already exists for the session, the transfer does not occur and an error message is generated.

Errors

If any of the following occur:

- The connection cannot be formed
- The user cannot log in
- The specified transfer parameters are not supported
- Moving to the specified remote directory fails

the command will fail and no files will be transferred.

References

generic formal function Remote_To_Local

function Ftp.Get_Set

function Ftp.Put

function Ftp.Put_Set

procedure Ftp.Send_Port

package Ftp_Defs

package Ftp_Name_Map

package Ftp_Profile

---

procedure Get_List

```
procedure Get_List
(Remote_File_List    : String                      := "";
 Local_Roof          : String                      := "$";
 Remote_Roof         : String                      := Ftp_Profile.Remote_Roof;
 Remote_Machine      : String                      :=
                                                   Ftp_Profile.Remote_Machine;
 Username            : String                      := Ftp_Profile.Username;
 Password            : String                      := Ftp_Profile.Password;
 Account             : String                      := Ftp_Profile.Account;
 Remote_Directory    : String                      :=
                                                 Ftp_Profile.Remote_Directory;
 Remote_Type         : Ftp_Name_Map.Machine_Type   := Ftp_Profile.Remote_Type;
 Append_To_File      : Boolean                     := False;
 Transfer_Type       : Ftp_Defs.Type_Code          :=
                                                    Ftp_Profile.Transfer_Type;
 Transfer_Mode       : Ftp_Defs.Mode_Code          :=
                                                    Ftp_Profile.Transfer_Mode;
 Transfer_Structure  : Ftp_Defs.Structure_Code     :=
                                               Ftp_Profile.Transfer_Structure;
 Send_Port           : Boolean                     :=
                                                Ftp_Profile.Send_Port_Enabled;
 Response            : Profile.Response_Profile    := Profile.Get);
```

Description

Retrieves multiple files from a remote host.

This procedure is used as a one-step operation to retrieve multiple files from a remote host. A connection is formed to the specified machine. The connection is logged in using the user information supplied. Files specified in the Remote_File_List parameter are retrieved. At the end of the transfer, the connection is disconnected.

The names of the local files are generated algorithmically from the names of the remote files, using the Remote_To_Local function. The Local_Roof, Remote_Roof, and Remote_Type parameters are used in this conversion.

Parameters

`Remote_File_List : String := "";`

Specifies the name of a local file. The file must contain text, with each line containing the name of one remote file.

`Local_Roof : String := "$";`

Specifies the parent for the file or files to be retrieved.

```
Remote_Roof : String := Ftp_Profile.Remote_Roof;
```

Specifies the parent on the remote machine for all files to be retrieved. If the null string is given, retrieved filenames are flattened; that is, the simple names of the source files are mapped to the simple names of the target files.

```
Remote_Machine : String := Ftp_Profile.Remote_Machine;
```

Specifies the machine to which the connection is formed and from which files are retrieved.

```
Username : String := Ftp_Profile.Username;
```

Specifies the username to be supplied to the remote server for login.

```
Password : String := Ftp_Profile.Password;
```

Specifies the password to be supplied to the remote server for login.

```
Account : String := Ftp_Profile.Account;
```

Specifies the account information supplied to the remote machine when it requires account information. The default value is obtained from the Account switch in the user's session switch file.

```
Remote_Directory : String := Ftp_Profile.Remote_Directory;
```

Specifies the remote directory context for this transfer. The value "" means that the Get_List procedure does not explicitly set the directory.

```
Remote_Type : Ftp_Name_Map.Machine_Type := Ftp_Profile.Remote_Type;
```

Specifies the machine type of the remote machine. This information is used when forming local filenames. The default is the value provided by the Remote_Type switch in the session switch file.

```
Append_To_File : Boolean := False;
```

Specifies, if true, that each retrieved file should be appended to the local file if it already exists. If the file does not exist, it is created. If false, the local file is overwritten if it exists and is created if it does not exist.

```
Transfer_Type : Ftp_Defs.Type_Code := Ftp_Profile.Transfer_Type;
```

Specifies the representation type to be used while transferring the files.

```
Transfer_Mode : Ftp_Defs.Mode_Code := Ftp_Profile.Transfer_Mode;
```

Specifies the transfer mode to be used while transferring the files.

```
Transfer_Structure  :  Ftp_Defs.Structure_Code  :=
                                         Ftp_Profile.Transfer_Structure;
```

Specifies the transfer structure to be used while transferring the files.

```
Send_Port  :  Boolean := Ftp_Profile.Send_Port_Enabled;
```

Specifies whether the server should be explicitly informed of the port to use for data transfer. It is recommended that this parameter be set to true (the default) when transferring multiple files in a single session.

```
Response  :  Profile.Response_Profile  := Profile.Get;
```

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

Restrictions

If a connection already exists for the session, the transfer does not occur and an error message is generated.

Errors

If the Remote_File_List cannot be opened or does not contain valid remote filenames, this command will not work.

If any of the following occur:

- The connection cannot be formed
- The user cannot log in
- The specified transfer parameters are not supported
- Moving to the specified remote directory fails

the command will fail and no files will be transferred.

If a file fails to transfer, the command may attempt to transfer the remaining files or terminate, depending on the specified Response_Profile (Profile.Reaction (Response)).

References

generic formal function Remote_To_Local

procedure Ftp.Get

procedure Ftp.Get_Set

procedure Ftp.Retrieve_List

procedure Ftp.Send_Port

---

procedure Get_Set

```
procedure Get_Set
(From_Remote_File_Set  : String                        := "";
 Local_Roof            : String                        := "$";
 Remote_Roof           : String                        :=
                                               Ftp_Profile.Remote_Roof;
 Remote_Machine        : String                        :=
                                            Ftp_Profile.Remote_Machine;
 Username              : String                        := Ftp_Profile.Username;
 Password              : String                        := Ftp_Profile.Password;
 Account               : String                        := Ftp_Profile.Account;
 Remote_Directory      : String                        :=
                                          Ftp_Profile.Remote_Directory;
 Remote_Type           : Ftp_Name_Map.Machine_Type     :=
                                              Ftp_Profile.Remote_Type;
 Append_To_File        : Boolean                       := False;
 Transfer_Type         : Ftp_Defs.Type_Code            :=
                                            Ftp_Profile.Transfer_Type;
 Transfer_Mode         : Ftp_Defs.Mode_Code            :=
                                            Ftp_Profile.Transfer_Mode;
 Transfer_Structure    : Ftp_Defs.Structure_Code       :=
                                       Ftp_Profile.Transfer_Structure;
 Send_Port             : Boolean                       :=
                                        Ftp_Profile.Send_Port_Enabled;
 Response              : Profile.Response_Profile      := Profile.Get);
```

Description

Retrieves multiple files from a remote host.

This procedure is used as a one-step operation to retrieve multiple files from a remote host. A connection is formed to the specified machine. The connection is logged in using the user information supplied. Files specified by the From_Remote_File_Set parameter are retrieved and placed in the context indicated by Local_Roof. At the end of the transfer, the connection is disconnected.

The names of the local files are generated algorithmically from the names of the remote files, using the Remote_To_Local function. The Local_Roof, Remote_Roof, and Remote_Type parameters are used in this conversion.

Parameters

`From_Remote_File_Set : String := "";`

Specifies the files to be retrieved from the remote host. The effect of the null string may vary for each implementation of the remote server. It is recommended that a nonnull argument be given. The wildcards used must be those of the remote machine.

```
Local_Roof : String := "$";
```

Specifies the parent for the file(s) to be retrieved.

```
Remote_Roof : String := Ftp_Profile.Remote_Roof;
```

Specifies the parent on the remote machine for all files to be retrieved. If the null string is given, the retrieved filenames are flattened; that is, the simple names of the source files are mapped to the simple names of the target files.

```
Remote_Machine : String := Ftp_Profile.Remote_Machine;
```

Specifies the machine to which the connection is formed and from which files are retrieved.

```
Username : String := Ftp_Profile.Username;
```

Specifies the username to be supplied to the remote server for login.

```
Password : String := Ftp_Profile.Password;
```

Specifies the password to be supplied to the remote server for login.

```
Account : String := Ftp_Profile.Account;
```

Specifies the account information supplied to the remote machine when it requires account information. The default value is obtained from the Account switch in the user's session switch file.

```
Remote_Directory : String := Ftp_Profile.Remote_Directory;
```

Specifies the remote directory context for this transfer. The value "" means that the Get_Set procedure does not explicitly set the directory.

```
Remote_Type : Ftp_Name_Map.Machine_Type := Ftp_Profile.Remote_Type;
```

Specifies the machine type of the remote machine. This information is used when forming local filenames. The default is the value provided by the Remote_Type switch in the session switch file.

```
Append_To_File : Boolean := False;
```

Specifies, if true, that each retrieved file should be appended to the local file if it already exists. If the file does not exist, it is created. If false, the local file is overwritten if it exists and is created if it does not exist.

```
Transfer_Type : Ftp_Defs.Type_Code := Ftp_Profile.Transfer_Type;
```

Specifies the representation type to be used while transferring the files.

```
Transfer_Mode : Ftp_Defs.Mode_Code := Ftp_Profile.Transfer_Mode;
```

Specifies the transfer mode to be used while transferring the files.

```
Transfer_Structure : Ftp_Defs.Structure_Code :=
                                        Ftp_Profile.Transfer_Structure;
```

Specifies the transfer structure to be used while transferring the files.

```
Send_Port : Boolean := Ftp_Profile.Send_Port_Enabled;
```

Specifies whether the server should be explicitly informed of the port to use for data transfer. It is recommended that this parameter be set to true (the default) when transferring multiple files in a single session.

```
Response : Profile.Response_Profile := Profile.Get;
```

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

Restrictions

If a connection already exists for the session, the transfer does not occur and an error message is generated.

Errors

The list of files to be retrieved is generated by requesting the remote server to produce a list of all files matching the specified template. If the remote server does not support this function, the Get_Set procedure will not work.

If any of the following occur:

- The connection cannot be formed
- The user cannot log in
- The specified transfer parameters are not supported
- Moving to the specified remote directory fails

the command will fail and no files will be transferred.

If a file fails to transfer, the command may attempt to transfer the remaining files or terminate, depending on the specified Response_Profile (Profile.Reaction (Response)).

procedure Get_Set
package !Tools.Networking.Transfer_Generic

References

generic formal function Remote_To_Local

procedure Ftp.Get

procedure Ftp.Put

procedure Ftp.Put_Set

procedure Ftp.Send_Port

package Ftp_Defs

package Ftp_Name_Map

package Ftp_Profile

generic formal function Local_To_Remote

```
with function Local_To_Remote (File_Name   : String;
                               Local_Roof  : String;
                               Remote_Roof : String;
                               Remote_Type : Ftp_Name_Map.Machine_Type)
                                                  return String;
```

Description

Returns the remote filename corresponding to the given local filename.

It is recommended that this function call the Ftp_Name_Map.Local_To_Remote function for any Remote_Type value it does not support. In this way, the instantiation of Transfer_Generic will be upwardly compatible with package Ftp.

Parameters

`File_Name : String;`

Specifies a local filename.

`Local_Roof : String;`

Specifies a roof directory on the local machine.

`Remote_Roof : String;`

Specifies a roof directory on the remote machine.

`Remote_Type : Ftp_Name_Map.Machine_Type;`

Specifies the type of the remote machine.

`return String;`

Returns a remote filename.

Errors

If this function raises an exception, the transfer of the named file will fail.

generic formal function Local_To_Remote
package !Tools.Networking.Transfer_Generic

References

function Ftp_Name_Map.Local_To_Remote

---

procedure Put

```
procedure Put
(From_Local_File      : String                         := "<IMAGE>";
 To_Remote_File       : String                         := "";
 Remote_Machine       : String                         :=
                                                        Ftp_Profile.Remote_Machine;
 Username             : String                         := Ftp_Profile.Username;
 Password             : String                         := Ftp_Profile.Password;
 Account              : String                         := Ftp_Profile.Account;
 Remote_Directory     : String                         :=
                                                      Ftp_Profile.Remote_Directory;
 Remote_Type          : Ftp_Name_Map.Machine_Type      := Ftp_Profile.Remote_Type;
 Append_To_File       : Boolean                        := False;
 Transfer_Type        : Ftp_Defs.Type_Code             :=
                                                        Ftp_Profile.Transfer_Type;
 Transfer_Mode        : Ftp_Defs.Mode_Code             :=
                                                        Ftp_Profile.Transfer_Mode;
 Transfer_Structure   : Ftp_Defs.Structure_Code        :=
                                                    Ftp_Profile.Transfer_Structure;
 Send_Port            : Boolean                        :=
                                                     Ftp_Profile.Send_Port_Enabled;
 Response             : Profile.Response_Profile       := Profile.Get);
```

Description

Stores a file onto a remote host.

This procedure is used as a one-step operation to store a file onto a remote host. A connection is formed to the specified machine. The connection is logged in using the user information supplied. The local file is transferred to the remote machine and placed in the remote file. At the end of the transfer, the connection is disconnected.

Parameters

`From_Local_File : String := "<IMAGE>";`

Specifies the local file to be transferred to the remote machine. It should be a nonnull argument. The default, "<IMAGE>", references the object associated with the highlighted area in which the cursor is located. If the cursor is not located in the highlighted area, "<IMAGE>" references the object associated with the window in which the cursor is located.

`To_Remote_File : String := "";`

Specifies the destination filename. If null, the filename is derived from the local filename, using the Local_To_Remote function.

```
Remote_Machine : String := Ftp_Profile.Remote_Machine;
```

Specifies the machine to which the connection is formed and onto which the file is stored.

```
Username : String := Ftp_Profile.Username;
```

Specifies the username to be supplied to the remote server for login.

```
Password : String := Ftp_Profile.Password;
```

Specifies the password to be supplied to the remote server for login.

```
Account : String := Ftp_Profile.Account;
```

Specifies the account information supplied to the remote machine when it requires account information. The default value is obtained from the Account switch in the user's session switch file.

```
Remote_Directory : String := Ftp_Profile.Remote_Directory;
```

Specifies the remote directory context for this transfer. The value "" means that the Put procedure does not explicitly set the directory.

```
Remote_Type : Ftp_Name_Map.Machine_Type := Ftp_Profile.Remote_Type;
```

Specifies the machine type of the remote machine. This information is used when forming remote filenames. The default is the value provided by the Remote_Type switch in the session switch file.

```
Append_To_File : Boolean := False;
```

Specifies, if true, that the stored file should be appended to the remote file if it already exists. If the file does not exist, it is created. If false, the remote file is overwritten if it exists and is created if it does not exist.

```
Transfer_Type : Ftp_Defs.Type_Code := Ftp_Profile.Transfer_Type;
```

Specifies the representation type to be used while transferring the file.

```
Transfer_Mode : Ftp_Defs.Mode_Code := Ftp_Profile.Transfer_Mode;
```

Specifies the transfer mode to be used while transferring the file.

```
Transfer_Structure : Ftp_Defs.Structure_Code :=
                                    Ftp_Profile.Transfer_Structure;
```

Specifies the transfer structure to be used while transferring the file.

```
Send_Port : Boolean := Ftp_Profile.Send_Port_Enabled;
```

Specifies whether the server should be explicitly informed of the port to use for data transfer. It is recommended that this parameter be set to true (the default) when transferring multiple files in a single session.

```
Response : Profile.Response_Profile := Profile.Get;
```

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

Restrictions

If a connection already exists for the session, the transfer does not occur and an error message is generated.

Errors

If any of the following occur:

- The connection cannot be formed
- The user cannot log in
- The specified transfer parameters are not supported
- Moving to the specified remote directory fails

the command will fail and no files will be transferred.

References

generic formal function Local_To_Remote

procedure Ftp.Get

procedure Ftp.Put_Set

procedure Ftp.Send_Port

procedure Put_Set

```
procedure Put_Set
(From_Local_File_Set  : String                         := "<IMAGE>";
 Local_Roof           : String                         := "$";
 Remote_Roof          : String                         :=
                                                        Ftp_Profile.Remote_Roof;
 Remote_Machine       : String                         :=
                                                     Ftp_Profile.Remote_Machine;
 Username             : String                         := Ftp_Profile.Username;
 Password             : String                         := Ftp_Profile.Password;
 Account              : String                         := Ftp_Profile.Account;
 Remote_Directory     : String                         :=
                                                   Ftp_Profile.Remote_Directory;
 Remote_Type          : Ftp_Name_Map.Machine_Type      :=
                                                        Ftp_Profile.Remote_Type;
 Append_To_File       : Boolean                        := False;
 Transfer_Type        : Ftp_Defs.Type_Code             :=
                                                      Ftp_Profile.Transfer_Type;
 Transfer_Mode        : Ftp_Defs.Mode_Code             :=
                                                      Ftp_Profile.Transfer_Mode;
 Transfer_Structure   : Ftp_Defs.Structure_Code        :=
                                                 Ftp_Profile.Transfer_Structure;
 Send_Port            : Boolean                        :=
                                                  Ftp_Profile.Send_Port_Enabled;
 Response             : Profile.Response_Profile       := Profile.Get);
```

Description

Stores multiple files onto a remote host.

This procedure is used as a one-step operation to store multiple files onto a remote host. A connection is formed to the specified machine. The connection is logged in using the user information supplied. Files specified by the From_Local_File_Set parameter are stored in the context indicated by the Remote_Roof parameter. At the end of the transfer, the connection is disconnected.

Parameters

`From_Local_File_Set : String := "<IMAGE>";`

Specifies files to be sent to the remote machine. The wildcards used must be those of the remote machine. The default, "<IMAGE>", references the object associated with the highlighted area in which the cursor is located. If the cursor is not located in the highlighted area, "<IMAGE>" references the object associated with the window in which the cursor is located.

`Local_Roof : String := "$";`

Specifies the ancestor directory for the file(s) to be transferred. If the null string is given, retrieved filenames are flattened; that is, the simple names of the source files are mapped to the simple names of the target files.

`Remote_Roof : String := Ftp_Profile.Remote_Roof;`

Specifies the ancestor directory on the remote machine for all files stored.

`Remote_Machine : String := Ftp_Profile.Remote_Machine;`

Specifies the machine to which the connection is formed and to which files are transferred.

`Username : String := Ftp_Profile.Username;`

Specifies the username to be supplied to the remote server for login.

`Password : String := Ftp_Profile.Password;`

Specifies the password to be supplied to the remote server for login.

`Account : String := Ftp_Profile.Account;`

Specifies the account information supplied to the remote machine when it requires account information. The default value is obtained from the Account switch in the user's session switch file.

`Remote_Directory : String := Ftp_Profile.Remote_Directory;`

Specifies the remote directory context for this transfer. The value "" means that the Put_Set procedure does not explicitly set the directory. The default is the null string ("").

`Remote_Type : Ftp_Name_Map.Machine_Type := Ftp_Profile.Remote_Type;`

Specifies the machine type of the remote machine. This information is used when forming remote filenames. The default is the value provided by the Remote_Type switch in the session switch file.

`Append_To_File : Boolean := False;`

Specifies, if true, that the transferred files should be appended to the destination files if they exist. If the files do not exist, they are created. If false, destination files are overwritten if they exist and are created if they do not exist.

`Transfer_Type : Ftp_Defs.Type_Code := Ftp_Profile.Transfer_Type;`

Specifies the representation type to be used while transferring the files.

```
Transfer_Mode : Ftp_Defs.Mode_Code := Ftp_Profile.Transfer_Mode;
```

Specifies the transfer mode to be used while transferring the files.

```
Transfer_Structure : Ftp_Defs.Structure_Code :=
                                             Ftp_Profile.Transfer_Structure;
```

Specifies the transfer structure to be used while transferring the files.

```
Send_Port : Boolean := Ftp_Profile.Send_Port_Enabled;
```

Specifies whether the server should be explicitly informed of the port to use for data transfer. It is recommended that this parameter be set to true (the default) when transferring multiple files in a single session.

```
Response : Profile.Response_Profile := Profile.Get;
```

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

Restrictions

If a connection already exists for the session, the transfers do not occur and an error message is generated.

Errors

If any of the following occur:

- The connection cannot be formed
- The user cannot log in
- The specified transfer parameters are not supported
- Moving to the specified remote directory fails

the command will fail and no files will be transferred.

If a file fails to transfer, the command may attempt to transfer the remaining files or terminate, depending on the specified Response_Profile (Profile.Reaction (Response)).

References

procedure Ftp.Get

procedure Ftp.Get_Set

procedure Ftp.Put

procedure Ftp.Send_Port

package Ftp_Defs

package Ftp_Name_Map

package Ftp_Profile

generic formal function Remote_To_Local

```
with function Remote_To_Local (File_Name   : String;
                               Local_Roof  : String;
                               Remote_Roof : String;
                               Remote_Type : Ftp_Name_Map.Machine_Type)
                                                     return String;
```

Description

Returns the local filename corresponding to the given remote filename.

It is recommended that this function call the Ftp_Name_Map.Remote_To_Local function for any Remote_Type value it does not support. In this way, the instantiation of Transfer_Generic will be upwardly compatible with package Ftp.

Parameters

`File_Name : String;`

Specifies a remote filename.

`Local_Roof : String;`

Specifies a roof directory on the local machine.

`Remote_Roof : String;`

Specifies a roof directory on the remote machine.

`Remote_Type : Ftp_Name_Map.Machine_Type;`

Specifies the type of the remote machine.

`return String;`

Returns a local filename.

Errors

If this function raises an exception, the transfer of the named file will fail.

References

function Ftp_Name_Map.Remote_To_Local

---

procedure Retrieve

```
procedure Retrieve
(From_Remote_File : String                     := "";
 To_Local_File    : String                     := "";
 Remote_Type      : Ftp_Name_Map.Machine_Type  :=
                                       Ftp_Profile.Current_Remote_Type;
 Append_To_File   : Boolean                    := False;
 Response         : Profile.Response_Profile   := Profile.Get;
 Account          : String                     := Ftp_Profile.Account);
```

Description

Retrieves a copy of a file from a remote machine.

The user session must already be connected and logged into the remote machine.

Parameters

`From_Remote_File : String := "";`

Specifies the remote file to be copied.

`To_Local_File : String := "";`

Specifies the local file into which the copy is to be written. If the null string is specified, the file is derived from the remote filename, using the Remote_To_Local function.

```
Remote_Type : Ftp_Name_Map.Machine_Type :=
                                       Ftp_Profile.Current_Remote_Type;
```

Specifies the machine type of the remote machine. This information is used in generating the local filenames, because some differences may exist in directory-naming and filenaming conventions. The default is the value provided by the Remote_Type switch in the session switch file.

`Append_To_File : Boolean := False;`

Specifies, if true, that the retrieved file should be appended to the local file if it exists. If the local file does not exist, it is created. If false, the local file is overwritten if it exists and is created if it does not exist.

`Response : Profile.Response_Profile := Profile.Get;`

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

```
Account : String := Ftp_Profile.Account;
```

Specifies the account information supplied to the remote machine when it requires account information. The default value is obtained from the Account switch in the user's session switch file.

Restrictions

The user first must have connected and logged into a remote server before issuing this command. If the session does not have an active FTP connection, an error message is produced and the command has no effect.

References

generic formal function Remote_To_Local

procedure Ftp.Connect

procedure Ftp.Login

procedure Ftp.Store

---

procedure Retrieve_List

```
procedure Retrieve_List
(Remote_File_List : String                        := "";
 Local_Roof       : String                        := "$";
 Remote_Roof      : String                        :=
                                      Ftp_Profile.Current_Remote_Roof;
 Remote_Type      : Ftp_Name_Map.Machine_Type     :=
                                      Ftp_Profile.Current_Remote_Type;
 Append_To_File   : Boolean                       := False;
 Response         : Profile.Response_Profile      := Profile.Get;
 Account          : String                        := Ftp_Profile.Account);
```

Description

Retrieves multiple files from a remote machine, using a list of remote filenames stored in a file on this machine.

The files are stored on the local machine with names mapped from the remote filenames. The names of the local files are generated algorithmically from the names of the remote files, using the Remote_To_Local function. The Local_Roof, Remote_Roof, and Remote_Type parameters are used in this conversion.

Parameters

```
Remote_File_List : String := "";
```

Specifies the name of a local file. The file must contain text, with each line containing the name of one remote file to be retrieved.

```
Local_Roof : String := "$";
```

Specifies the local parent of the files to be retrieved.

```
Remote_Roof : String := Ftp_Profile.Current_Remote_Roof;
```

Specifies the parent for the files at the remote machine. If the null string is given, retrieved filenames are flattened; that is, the simple names of the source files are mapped to the simple names of the target files.

```
Remote_Type : Ftp_Name_Map.Machine_Type :=
                                   Ftp_Profile.Current_Remote_Type;
```

Specifies the machine type of the remote machine. This information is used in generating the local filenames, because some differences may exist in directory-naming and filenaming conventions. The default is the value provided by the Remote_Type switch in the session switch file.

```
Append_To_File : Boolean := False;
```

Specifies, if true, that retrieved files should be appended to local files if they already exist. If the local files do not exist, they are created. If false, the local files are overwritten if they exist and are created if they do not exist.

```
Response : Profile.Response_Profile := Profile.Get;
```

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

```
Account : String := Ftp_Profile.Account;
```

Specifies the account information supplied to the remote machine when it requires account information. The default value is obtained from the Account switch in the user's session switch file.

Restrictions

The user first must have connected and logged into a remote server before issuing this command. If the session does not have an active FTP connection, an error message is produced and the command has no effect.

Errors

If the Remote_File_List file cannot be opened or does not contain valid remote filenames, this command does not work.

If a file fails to transfer, the command may attempt to transfer the remaining files or terminate, depending on the specified Response_Profile (Profile.Reaction (Response)).

References

generic formal function Remote_To_Local

procedure Ftp.Get_List

procedure Ftp.Retrieve

procedure Retrieve_Set

```
procedure Retrieve_Set
(From_Remote_File_Set : String                      := "";
 Local_Roof           : String                      := "$";
 Remote_Roof          : String                      :=
                                    Ftp_Profile.Current_Remote_Roof;
 Remote_Type          : Ftp_Name_Map.Machine_Type  :=
                                    Ftp_Profile.Current_Remote_Type;
 Append_To_File       : Boolean                     := False;
 Response             : Profile.Response_Profile    := Profile.Get;
 Account              : String                      := Ftp_Profile.Account);
```

Description

Retrieves multiple files from a remote machine.

The names of the local files are generated algorithmically from the names of the remote files, using the Remote_To_Local function. The Local_Roof, Remote_Roof, and Remote_Type parameters are used in this conversion.

Parameters

`From_Remote_File_Set : String := "";`

Specifies a file template for the files to be retrieved. The wildcards used must be those of the remote machine.

`Local_Roof : String := "$";`

Specifies the local parent of the files to be retrieved.

`Remote_Roof : String := Ftp_Profile.Current_Remote_Roof;`

Specifies the parent for the files at the remote machine. If the null string is given, retrieved filenames are flattened; that is, the simple names of the source files are mapped to the simple names of the target files.

```
Remote_Type : Ftp_Name_Map.Machine_Type :=
                                   Ftp_Profile.Current_Remote_Type;
```

Specifies the machine type of the remote machine. This information is used in generating the local filenames, because some differences may exist in directory-naming and filenaming conventions. The default is the value provided by the Remote_Type switch in the session switch file.

```
Append_To_File : Boolean := False;
```

Specifies, if true, that retrieved files should be appended to local files if they already exist. If the local files do not exist, they are created. If false, the local files are overwritten if they exist and are created if they do not exist.

```
Response : Profile.Response_Profile := Profile.Get;
```

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

```
Account : String := Ftp_Profile.Account;
```

Specifies the account information supplied to the remote machine when it requires account information. The default value is obtained from the Account switch in the user's session switch file.

Restrictions

The user first must have connected and logged into a remote server before issuing this command. If the session does not have an active FTP connection, an error message is produced and the command has no effect.

Errors

The list of files to be retrieved is generated by requesting the remote server to produce a list of all files matching the specified template. If the remote server does not support this function, the Retrieve_Set procedure does not work.

If a file fails to transfer, the command may attempt to transfer the remaining files or terminate, depending on the specified Response_Profile (Profile.Reaction (Response)).

References

generic formal function Remote_To_Local

procedure Ftp.Get_Set

procedure Ftp.Retrieve_List

procedure Store

```
procedure Store
(From_Local_File  : String                       := "<IMAGE>";
 To_Remote_File   : String                       := "";
 Remote_Type      : Ftp_Name_Map.Machine_Type    :=
                                     Ftp_Profile.Current_Remote_Type;
 Append_To_File   : Boolean                      := False;
 Response         : Profile.Response_Profile     := Profile.Get;
 Account          : String                       := Ftp_Profile.Account);
```

Description

Stores a copy of the local file onto the remote machine to which the session is currently connected.

The transfer is conducted in accordance with all other separately specified parameters (Text, Image, Block, and so on).

The user session must already be connected and logged into the remote machine.

Parameters

```
From_Local_File : String := "<IMAGE>";
```

Specifies the local file to be copied to the remote machine. The default, "<IMAGE>", references the object associated with the highlighted area in which the cursor is located. If the cursor is not located in the highlighted area, "<IMAGE>" references the object associated with the window in which the cursor is located.

```
To_Remote_File : String := "";
```

Specifies the destination name for the file on the remote machine. If the null string is specified, the remote filename is derived from the local filename, using the Local_To_Remote function.

```
Remote_Type : Ftp_Name_Map.Machine_Type :=
                                     Ftp_Profile.Current_Remote_Type;
```

Specifies the machine type of the remote machine. This information is used in generating the remote filenames, because some differences may exist in directory-naming and filenaming conventions. The default is the value provided by the Remote_Type switch in the session switch file.

```
Append_To_File : Boolean := False;
```

Specifies, if true, that the file should be appended to the remote file if it already exists. If the remote file does not exist, it is created. If false, the remote file is overwritten if it exists and is created if it does not exist.

```
Response : Profile.Response_Profile := Profile.Get;
```

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

```
Account : String := Ftp_Profile.Account;
```

Specifies the account information supplied to the remote machine when it requires account information. The default value is obtained from the Account switch in the user's session switch file.

Restrictions

The user first must have connected and logged into a remote server before issuing this command. If the session does not have an active FTP connection, an error message is produced and the command has no effect.

References

generic formal function Local_To_Remote

procedure Ftp.Retrieve

procedure Store_Set

```
procedure Store_Set
(From_Local_File_Set  : String                      := "<IMAGE>";
 Local_Roof           : String                      := "$";
 Remote_Roof          : String                      :=
                                         Ftp_Profile.Current_Remote_Roof;
 Remote_Type          : Ftp_Name_Map.Machine_Type   :=
                                         Ftp_Profile.Current_Remote_Type;
 Append_To_File       : Boolean                     := False;
 Response             : Profile.Response_Profile    := Profile.Get;
 Account              : String                      := Ftp_Profile.Account);
```

Description

Stores multiple files from the local machine onto the currently connected remote machine.

The transfer is conducted in accordance with all other separately specified parameters (Text, Image, Block, and so on).

Parameters

`From_Local_File_Set : String := "<IMAGE>";`

Specifies a file template for the local file to be stored. The wildcards used must be those of the remote machine. The default, "<IMAGE>", references the object associated with the highlighted area in which the cursor is located. If the cursor is not located in the highlighted area, "<IMAGE>" references the object associated with the window in which the cursor is located.

`Local_Roof : String := "$";`

Specifies the local ancestor of the files to be stored onto the remote machine. If the null string is given, retrieved filenames are flattened; that is, the simple names of the source files are mapped to the simple names of the target files.

`Remote_Roof : String := Ftp_Profile.Current_Remote_Roof;`

Specifies the ancestor for the files at the remote machine.

```
Remote_Type : Ftp_Name_Map.Machine_Type :=
                                     Ftp_Profile.Current_Remote_Type;
```

Specifies the machine type of the remote machine. This information is used in generating the remote filenames, because some differences may exist in directory-naming and filenaming conventions. The default is the value provided by the Remote_Type switch in the session switch file.

```
Append_To_File : Boolean := False;
```

Specifies, if true, that files should be appended to the remote files if they already exist. If the remote files do not exist, they are created. If false, the files are overwritten if they exist and are created if they do not exist.

```
Response : Profile.Response_Profile := Profile.Get;
```

Specifies how to respond to errors, how to generate logs, and what activities and switches to use during the execution of this command. The default is the current job response profile.

```
Account : String := Ftp_Profile.Account;
```

Specifies the account information supplied to the remote machine when it requires account information. The default value is obtained from the Account switch in the user's session switch file.

Restrictions

The user first must have connected and logged into a remote server before issuing this command. If the session does not have an active FTP connection, an error message is produced and the command has no effect.

Errors

If a file fails to transfer, the command may attempt to transfer the remaining files or terminate, depending on the specified Response_Profile (Profile.Reaction (Response)).

References

procedure Ftp.Retrieve

procedure Ftp.Retrieve_Set

procedure Ftp.Store

package Ftp_Profile

end Transfer_Generic;

RATIONAL

Index

This index contains entries for each unit and its declarations as well as definitions, topical cross-references, exceptions raised, errors, enumerations, pragmas, switches, and the like. The entries for each unit are arranged alphabetically by simple name. An italic page number indicates the primary reference for an entry.

RATIONAL

RATIONAL

READER'S COMMENTS

Note: This form is for documentation comments only. You can also submit problem reports and comments electronically by using the SIMS problem-reporting system. If you use SIMS to submit documentation comments, please indicate the manual name, book name, and page number.

Did you find this book understandable, usable, and well organized? Please comment and list any suggestions for improvement.

If you found errors in this book, please specify the error and the page number. If you prefer, attach a photocopy with the error marked.

Indicate any additions or changes you would like to see in the index.

How much experience have you had with the Rational Environment?

6 months or less _______ 1 year _______ 3 years or more ______

How much experience have you had with the Ada programming language?

6 months or less _______ 1 year _______ 3 years or more ______

Name (optional)________________________ Date__________
Company ________________________
Address ________________________
City ______________ State __________ ZIP Code __________

Please return this form to:

Publications Department
Rational
1501 Salado Drive
Mountain View, CA 94043

Rational Networking—TCP/IP Reference Manual

Master Index: Networking

Document Control Number: 8003A

Rev. 1.0, July 1987 (Delta)

Note the Reader's Comments form on the last page of this book, which requests the user's evaluation to assist Rational in preparing future documentation.

Rational
1501 Salado Drive
Mountain View, California 94043

Master Index

The Master Index combines the indexes for the four books in the *Rational Networking—TCP/IP Reference Manual.* Thus, the prefix to the page number is the abbreviation for the book in which the item can be found. This index contains entries for each unit and its declarations as well as definitions, topical cross-references, exceptions raised, errors, enumerations, pragmas, switches, and the like. The entries for each unit are arranged alphabetically by simple name. An italic page number indicates the primary reference for an entry.

RATIONAL

READER'S COMMENTS

Note: This form is for documentation comments only. You can also submit problem reports and comments electronically by using the SIMS problem-reporting system. If you use SIMS to submit documentation comments, please indicate the manual name, book name, and page number.

Did you find this book understandable, usable, and well organized? Please comment and list any suggestions for improvement.

____

If you found errors in this book, please specify the error and the page number. If you prefer, attach a photocopy with the error marked.

____

Indicate any additions or changes you would like to see in the index.

____

How much experience have you had with the Rational Environment?

6 months or less _______ 1 year _______ 3 years or more _______

How much experience have you had with the Ada programming language?

6 months or less _______ 1 year _______ 3 years or more _______

Name (optional)____________________ Date__________
Company ______________________________
Address ______________________________
City ____________ State ____________ ZIP Code ____________

Please return this form to: **Publications Department**
Rational
1501 Salado Drive
Mountain View, CA 94043