Gentran:Server® for Windows®

Script Language Reference Guide

Version 5.1

Sterling Commerce
An IBM Company

4232-510-USER06-0000

Copyright Notice

Gentran:Server for Windows

Sterling Commerce Software
Trade Secret Notice

THE GENTRAN:SERVER FOR WINDOWS SOFTWARE ("STERLING COMMERCE SOFTWARE") IS THE CONFIDENTIAL AND TRADE SECRET PROPERTY OF STERLING COMMERCE, INC., ITS AFFILIATED COMPANIES OR ITS OR THEIR LICENSORS, AND IS PROVIDED UNDER THE TERMS OF A LICENSE AGREEMENT. NO DUPLICATION OR DISCLOSURE WITHOUT PRIOR WRITTEN PERMISSION. RESTRICTED RIGHTS.

This documentation, the Sterling Commerce Software it describes, and the information and know-how they contain constitute the proprietary, confidential and valuable trade secret information of Sterling Commerce, Inc., its affiliated companies or its or their licensors, and may not be used for any unauthorized purpose, or disclosed to others without the prior written permission of the applicable Sterling Commerce entity. This documentation and the Sterling Commerce Software that it describes have been provided pursuant to a license agreement that contains prohibitions against and/or restrictions on their copying, modification and use. Duplication, in whole or in part, if and when permitted, shall bear this notice and the Sterling Commerce, Inc. copyright notice.

As and when provided to any governmental entity, government contractor or subcontractor subject to the FARs, this documentation is provided with RESTRICTED RIGHTS under Title 48 CFR 52.227-19. Further, as and when provided to any governmental entity, government contractor or subcontractor subject to DFARs, this documentation and the Sterling Commerce Software it describes are provided pursuant to the customary Sterling Commerce license, as described in Title 48 CFR 227-7202 with respect to commercial software and commercial software documentation.

These terms of use shall be governed by the laws of the State of Ohio, USA, without regard to its conflict of laws provisions. If you are accessing the Sterling Commerce Software under an executed agreement, then nothing in these terms and conditions supersedes or modifies the executed agreement.

Product names mentioned herein may be trademarks and/or registered trademarks of their respective companies. Gentran and Gentran:Server are registered trademarks of Sterling Commerce, Inc.

Third Party Software:
Portions of the Sterling Commerce Software may include products, or may be distributed on the same storage media with products, ("Third Party Software") offered by third parties ("Third Party Licensors").

Warranty Disclaimer
This documentation and the Sterling Commerce Software which it describes are licensed either "AS IS" or with a limited warranty, as set forth in the Sterling Commerce license agreement. Other than any limited warranties provided, NO OTHER WARRANTY IS EXPRESSED AND NONE SHALL BE IMPLIED, INCLUDING THE WARRANTIES OF MERCHANTABILITY AND FITNESS FOR USE OR FOR A PARTICULAR PURPOSE. The applicable Sterling Commerce entity reserves the right to revise this publication from time to time and to make changes in the content hereof without the obligation to notify any person or entity of such revisions or changes.

The Third Party Software is provided 'AS IS' WITHOUT ANY WARRANTY AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. FURTHER, IF YOU ARE LOCATED OR ACCESSING THIS SOFTWARE IN THE UNITED STATES, ANY EXPRESS OR IMPLIED WARRANTY REGARDING TITLE OR NON-INFRINGEMENT ARE DISCLAIMED.

Sterling Commerce, Inc.
4600 Lakehurst Court Dublin, OH 43016-2000 * 614/793-7000

Table of Contents

About This Guide

Contents

Introduction

Overview

This guide describes the script language provided for use with the Gentran:Server for Windows communications subsystem.

Intended audience

The intended audience for this document is:

- Gentran:Server system administrators
- advanced Gentran:Server for Windows users

Prerequisite knowledge

The audience using this software should be familiar with:

- Microsoft® Windows
- Gentran:Server for Windows
- Communications protocols

Description of Contents

Introduction

This section describes how this manual is organized.

Organization of chapters

This guide is organized into chapters. A brief description of each chapter's contents follows.

- *About this Guide* explains the content and organization of this guide.
- *Script Language Overview* provides script language command overview information.
- *Script Language Reference* provides an alphabetical listing of all script language commands.

Online Help

Introduction

The majority of the documentation in this manual is contained in the Communications Gateway and Advanced Data Distribution Online Help systems.

Getting Support

Introduction

The Sterling Commerce Gentran:Server software is supported by trained product support personnel who are available to help you with product questions or concerns.

Note
Gentran:Server Customer Support does not support non-Sterling Commerce products (e.g., SQL Server, Oracle, etc.), but can assist you in configuring non-Sterling Commerce products to work with Gentran:Server.

Phone number

For assistance, please refer to your *Getting Started Guide* to determine which support phone number you should use.

Before calling support

To help us provide prompt service, we ask that you do the following:

- Attempt to recreate any problem that you encounter and record the exact sequence of events.
- When you call product support, you should be prepared to provide us with the information below.

Information	Description
Identification	Your company name, your name, telephone number and extension, and the case number (if the question refers to a previously reported issue).
System Configuration	The Gentran:Server version (and any service packs installed) and information about the primary Gentran system controller and all machines experiencing problems, including: the Windows operating system version, amount of memory, available disk space, database version, Microsoft Data Access (MDAC) version, and Internet Explorer version. Also, please describe any recent changes in your hardware, software, or the configuration of your system.
System Data Store	Which machines contain folders in the system data store?
Error Messages	Record the exact wording of any error messages you receive and the point in the software where the error occurred, as well as any log files.
Attempted Solutions	Record any steps that you took attempting to resolve the problem and note all the outcomes, and provide an estimate on how many times the problem occurred and whether it can be reproduced.

Accessing the Sterling Commerce Support Web Site

The Sterling Commerce Customer Support Web Site contains valuable information about getting support for Gentran:Server for Windows, including the:

- scope of support services
- customer support policies
- call prioritizing
- customer support phone directory
- how to create new Support on Demand cases
- how to check the status of Support on Demand cases
- how to add information to Support on Demand cases

The Customer Support Web Site is constantly updated and all Sterling Commerce customers have access to it. This web site also contains the most recent product updates and is a valuable source of product information.

Reference
Refer to the *Getting Started Guide* for information on how to access the Customer Support Web Site.

Documentation

The Customer Support Web Site contains a documentation library, which has the entire Gentran:Server for Windows documentation set. You can download the product manuals in PDF format from this library at any time.

CHAPTER

1

Script Language Overview

Contents

Overview

In this chapter

This chapter describes the script language overview information.

Format of command reference pages

Each command is shown using the following format:

- Command format—a description of the command showing the format to be used in a script.
- Explanation of command—a brief description of the command.
- Parameters—a list of the parameters used in the command (if applicable).
- Return value—a list of the possible return values (if applicable).
- Example—a brief example showing how the command is used in a script.

Format of commands

Commands are shown using the following format:

```
CommandName(string Required_Parameter,
[integer Optional_Parameter])
```

Each command is followed by zero, one, or more parameters. Parameters are enclosed in parentheses ( ) and separated by commas (,). Parameter names are case sensitive. If a command does not have any parameters, it is shown with empty parentheses.

The parameters of each command are shown with the data type of the parameters. In the example above, `Required_Parameter` is defined as a string parameter and would be used to hold one or more printable characters. `Optional_Parameter` is defined as an integer parameter and would be used to hold whole numbers that do not have decimal fractions.

Types of parameters

Commands may have two types of parameters:

- optional parameters—shown inside square brackets ([ ])
- required parameters—shown without square brackets

Declaring parameters

All parameters in the script commands must be declared before you can use them in the script. A declaration consists of a data type and the parameter which has that type.

Examples

```
string StopRcv;
integer TimeOut;
array line[10];
```

Data types

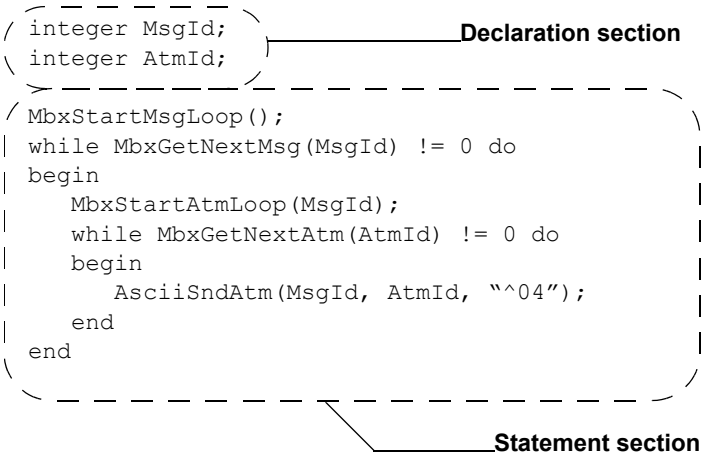
Parameters can have the following data types:

- Integer—a whole number without a decimal component
- Real—a number that may have a decimal component
- String—one or more printable characters
- Datetime—a date or time
- Array—a table of multiple occurrences of a single data type

Format of scripts

Scripts are divided into two sections: a declaration section used to define the parameters used in the script, and a statements section which holds the actual script commands.

Example



Keywords

A keyword is a special command used in the script to control the flow of the script. Keywords are used in conditional logic to test for matches to defined values.

Example

```
if RcvResult = 1 then
  LogMessage("receive no data condition occurred");
```

In this example a log message is generated when the value of `RcvResult` is equal to the number 1. The **if…then** keywords control the script such that if the value of `RcvResult` does not equal 1, no log message is generated.

Available keywords

The following keywords are available for use in the communications scripts:

- if…then…else
- while…do
- begin…end
- continue…break

Expressions

An expression is a logical unit that the system evaluates.

Examples

```
RcvResult = 1;
MbxGetNextMsg(MsgId) != 0;
A + B
```

Operators

Operators define the simplest operation in an expression. This table lists the operators used in script commands:

Operator	Description
+	addition, concatenation
-	subtraction
*	multiplication
/	division
=	assignment, equality
>	greater-than
<	less-than
>=	greater-than or equal to
<=	less-than or equal to
!=	not equal to
!	logical not
&	logical and
|	logical or
<<	date modification

Line terminators

Declarations and statements are terminated with a semi-colon (;).

Exception

Keyword statements (like **if…then** or **while…do**) are not terminated with a semi-colon. Only the statements within these keyword statements are terminated with a semi-colon.

Example

```
if RcvResult = 1 then
  LogMessage("receive no data condition occurred");
else if RcvResult = 2 then
  LogMessage("receive error condition occurred");
```

CHAPTER

2

Script Language Reference

Contents

AnsiClearRcvMsg

Command format	`AnsiClearRcvMsg();`
Explanation of command	This command is used to receive data using the AnsiClear protocol. Data received is stored in a mailbox message with one attachment.
Parameters	There are no parameters for this command.
Return values	There are no return values for this command.

AnsiClearSetRcvRsp

Command format

```
AnsiClearSetRcvRsp(string Response);
```

Explanation of command

This command sets the response to send when receiving data. Not all systems require a response and the default is to not send any response.

Parameters

Response	Contains response to send. This can be one or more characters.

Return value

There are no return values for this command.

Example

```
AnsiClearSetRcvRsp("^0D");
```

AnsiClearSetSndRsp

Command format

```
AnsiClearSetSndRsp(string Response);
```

Explanation of command

This command sets the response to expect when sending data. Not all systems will send a response and the default is not to expect any response.

Parameters

Response	Contains the response to expect. This can be one more more characters

Return value

There are no return values for this command.

Example

```
AnsiClearSetSndRsp("^0D");
```

AnsiClearSetTerm

Command format

```
AnsiClearSetTerm(string Terminator);
```

Explanation of command

This command sets the termination character to be used for sending and receiving. The default is an ASCII carriage return (0x0D).

Parameters

Terminator Contains the termination character.

Return value

There are no return values for this command.

Example

```
AnsiClearSetTerm("^0D");
```

AnsiClearSndAtm

Command format

```
AnsiClearSndAtm(integer MsgId, integer AtmId);
```

Explanation of command

This command is used to send a mailbox attachment using the AnsiClear protocol.

Parameters

MsgId	Must contain the mailbox message identifier returned from **MbxGetNextMsg**.
AtmId	Must contain the mailbox attachment identifier returned from **MbxGetNextAtm**.

Return value

There are no return values for this command.

Example

```
integer MsgId;
integer AtmId;
MbxStartMsgLoop();
while MbxGetNextMsg(MsgId) != 0 do
begin
   MbxStartAtmLoop(MsgId);
   while MbxGetNextAtm(AtmId) != 0 do
   begin
      AnsiClearSndAtm(MsgId, AtmId);
   end
end
```

AsciiRcvCtl

Command format

```
AsciiRcvCtl(string StopRcv, [integer Timeout],
[integer ContinueOnTimeout])
AsciiRcvCtl(integer ByteCount, [integer Timeout], [integer
ContinueOnTimeout])
```

Explanation of command

This command receives data using the ASCII protocol until the specified string or bytecount is found.

Parameters

StopRcv	Contains the control string used to indicate when to stop receiving.
ByteCount	Contains the number of bytes to receive.
Timeout	Optional. Defaults to 60 seconds. Used to override the default.
ContinueOnTimeout	Optional. Defaults to 0. Set to 1 if you wish the script to continue processing even though a timeout occurred before the stop receiving condition.

Return value

There are no return values for this command.

Example

```
//Receive data until the string "hello" is received
AsciiRcvCtl("hello");
AsciiRcvCtl("hello", 100);
AsciiRcvCtl("hello", 100, 1);
//Receive data 10 bytes of data
AsciiRcvCtl(10);
AsciiRcvCtl(10, 100);
AsciiRcvCtl(10, 100, 1);
```

AsciiRcvFile

Command format

```
integer AsciiRcvFile(string FileSpec, string EndOfFile, [integer
Terminate]);
```

Explanation of command

This command is used to receive data using the ASCII protocol. This command can be used in conjunction with **SetRcvNoData** and **SetRcvError** to look for other data responses in place of or lack of a receive file.

Parameters

FileSpec	A fully qualified filespec pointing to a file. UNC naming can be used.
EndOfFile	Used to denote when end of file has been reached.
Terminate	Optional. Defaults to 0. If set to 1, will terminate the receive if either the RcvNoData or RcvError conditions occur.

Return value

0	If end of file condition occurred.
1	If the RcvNoData condition occurred.
2	If the RcvError condition occurred.

Example

```
integer RcvResult;
SetRcvNoData("no data");
SetRcvError("error");
RcvResult = AsciiRcvFile("c:\temp\rcv.txt", "^04");
if RcvResult = 1 then
  LogMessage("receive no data condition occurred");
else if RcvResult = 2 then
  LogMessage("receive error condition occurred");
```

AsciiRcvMsg

Command format

```
integer AsciiRcvMsg(string EndOfFile,
[integer Terminate]);
```

Explanation of command

This command is used to receive data using the ASCII protocol. Data received is stored in a mailbox message with one attachment. This command can be used in conjunction with **SetRcvNoData** and **SetRcvError** to look for other data responses in place of or lack of a receive file.

Parameters

EndOfFile	Used to denote when end of file has been reached.
Terminate	Optional. Defaults to 0. If set to 1, will terminate the receive if either the **RcvNoData** or **RcvError** conditions occur.

Return value

0	If end of file condition occurred.
1	If the **RcvNoData** condition occurred.
2	If the **RcvError** condition occurred.

Example

```
integer RcvResult;
SetRcvNoData("no data");
SetRcvError("error");
RcvResult = AsciiRcvMsg("^04");
if RcvResult = 1 then
  LogMessage("receive no data condition occurred");
else if RcvResult = 2 then
  LogMessage("receive error condition occurred");
```

AsciiSndAtm

Command format

```
AsciiSndAtm(integer MsgId, integer AtmId,
string EndOfFile);
```

Explanation of command

This command is used to send a mailbox attachment using the ASCII protocol.

Parameters

MsgId	Must contain the mailbox message identifier returned from **MbxGetNextMsg.**
AtmId	Must contain the mailbox attachment identifier returned from **MbxGetNextAtm.**
EndOfFile	Contains any data to be sent to denote end of file.

Return value

There are no return values for this command.

Example

```
integer MsgId;
integer AtmId;
MbxStartMsgLoop();
while MbxGetNextMsg(MsgId) != 0 do
begin
   MbxStartAtmLoop(MsgId);
   while MbxGetNextAtm(AtmId) != 0 do
   begin
      AsciiSndAtm(MsgId, AtmId, "^04");
   end
end
```

AsciiSndCtl

Command format

```
AsciiSndCtl(string Data);
```

Explanation of command

This command transmits the specified string using ASCII protocol.

Parameters

Data	Contains the control string to be transmitted.

Return value

There are no return values for this command.

Example

```
AsciiSndCtl("hello world");
```

AsciiSndFile

Command format

```
AsciiSndFile(string FileSpec, string EndOfFile);
```

Explanation of command

This command is used to send a file using the ASCII protocol.

Parameters

FileSpec	A fully qualified filespec pointing to a file. UNC naming can be used.
EndOfFile	Contains any data to be sent to denote end of file.

Return value

There are no return values for this command.

Example

```
integer MsgId;
integer AtmId;
AsciiSndFile("c:\temp\snd.txt", "^04");
```

atoi

Command format

```
integer_variable = atoi(string);
```

Explanation of command

The **atoi** function is a numerical function that converts strings into integers. The numerical functions enable you to convert one data type to another.

Parameters

string	string variable

Return values

This function returns the integer value of a string.

Example

```
integer a;
string[20] s;
s = "5";
a = atoi(s);
// "a" contains the value 5
```

aton

Command format

```
real = aton(string);
```

Explanation of command

The **aton** function is a numerical function that converts strings into real numbers. The numerical functions enable you to convert one data type to another.

Parameters

string	The string to be converted into a real number.

Return values

This function returns the real value of a string.

Example

```
real a;
string[20] s;
s = "3.14159";
a = aton(s);
// "a" contains the value 3.14159
```

Begin…End

Command format

```
begin
statement1;
statement2;
end
```

Explanation of command

Begin…end keywords are used to contain multiple statements in the body of **if…then,** or **while…do** loops.

Note
You may omit the **begin** and **end** keywords if you only use a single statement in the loop.

Parameters

There are no parameters for this command.

Return value

There are no return values for this command.

Example

```
while MbxGetNextAtm(AtmId) != 0 do
   begin
      MbxGetAtmFileName(MsgId, AtmId, FileName);
      FtpSndAtm(MsgId, AtmId, FileName);
   end
```

BisyncAutoAnswer

Command format

```
BisyncAutoAnswer();
```

Explanation of command

This command places the bisynchronous device into an auto answer mode waiting for incoming calls. This command should only be used in a host pool script for bisynchronous devices.

Parameters

There are no parameters for this command.

Return value

There are no return values for this command.

BisyncClose

Command format

```
BisyncClose();
```

Explanation of command

This command should be issued after the completion of all **BisyncRead** or **BisyncWrite** commands to close the current transaction.

Parameters

There are no parameters for this command.

Return value

There are no return values for this command.

Example

```
BisyncOpen(OPEN_READ_TEXT);
BisyncRcvMsg();
BisyncClose();
```

BisyncConfig

Command format

```
BisyncConfig(string ConfigFile);
```

Explanation of command

This command allows you to load a configuration file that overrides the current configuration file.

Parameters

ConfigFile Contains the new configuration filename.

Return value

There are no return values for this command.

BisyncDial

Command format

```
BisyncDial();
```

Explanation of command

This command dials the phone number stored in the properties. This command allows you to load a new configuration file (by using the **BisyncConfig**) before dialing. If the system dialed automatically before calling **BisyncConfig**, the line would be dropped.

Parameters

There are no parameters for this command.

Return value

There are no return values for this command.

Example

```
BisyncConfig("NewConfig");
BisyncDial();
```

BisyncEot

Command format

```
BisyncEot();
```

Explanation of command

This command allows you to unconditionally clear the line by sending an end of transmission character and is usually preceded by a **BisyncClose** command.

Parameters

There are no parameters for this command.

Return value

There are no return values for this command.

Example

```
BisyncOpen(WRITE_TEXT);
BisyncSndCtl("Logon");
BisyncClose();
BisyncEot();
```

BisyncOpen

Command format

```
BisyncOpen(integer Mode);
```

Explanation of command

This command opens the line for sending or receiving data based on which constant is specified. When opening the line for receive, it waits for a line bid and responds with an acknowledgement. When opening the line for send, it bids the line and waits for an acknowledgement.

Parameters

Mode Contains one of the following open mode constants:

Constant	Description
READ_TEXT	Allows receiving non-transparent text data with translation. Transparent binary data is received without translation.
READ_NO_TRANSLATE	Allows receiving non-transparent text data without translation.
READ_BINARY_TRANSLATE	Allows receiving transparent binary data with translation.
WRITE_TEXT	Allows sending non-transparent text data with translation.
WRITE_BINARY	Allows sending transparent binary data without translation.
WRITE_TEXT_NO_TRANSLATE	Allows sending non-transparent text data without translation.
WRITE_NO_IRS	Allows sending non-transparent text data with translation but no record separators are included. This essentially packs the data into one record.
WRITE_BINARY_TRANSLATE	Allows sending transparent binary data with translation.
WRITE_TEXT_BINARY	Allows sending transparent binary data with translation but with special translation of special characters (CR/LF to RS).

Return value There are no return values for this command.

Example

```
BisyncOpen(READ_TEXT);
BisyncRcvMsg();
BisyncClose();
```

BisyncRcvCtl

Command format

```
Integer BisyncRcvCtl(integer EndRcv);
```

Explanation of command

This command is used to receive data into an internal receive buffer that can be used to check its contents. It specifies other protocol characters that indicate when to stop receiving and then returns the value for the protocol character that ended the receive.

Parameters

EndRcv	Specifies a protocol character that indicates when to stop receiving.
1	Stops receiving data when an EOT is received.
2	Stops receiving data when an ETB is received.
3	Stops receiving data when an ETX is received.
4	Stops receiving data when an ETB or ETX is received. The return code depends on what ended the receive.

Return value

1	An EOT was received.
2	An ETB was received.
3	An ETX was received.

Example

```
integer iResult;
iResult = BisyncRcvCtl(4);
if iResult = 1 then
   LogMessage("got EOT");
if iResult = 2 then
   LogMessage("got ETB");
if iResult = 3 then
   LogMessage("got ETX");
if RcvBufSearch("hello") = 1 then
   LogMessage("got hello");
```

Related topic

See the BisyncRcvCtl (integer EndRcv) script for an additional form of this command.

BisyncRcvFile

Command format

```
integer BisyncRcvFile(string FileSpec);
```

Explanation of command

This command is used to receive a file using the Bisync protocol. This command can be used in conjunction with **SetRcvNoData** and **SetRcvError** to look for other data responses in place of or lack of a receive file.

Parameters

FileSpec	A fully qualified filespec pointing to a file. UNC naming can be used.

Return value

0	Normal end of file.
1	If the RcvNoData condition occurred.
2	If the RcvError condition occurred.

Example

```
integer RcvResult;
SetRcvNoData("no data");
SetRcvError("error");
RcvResult = BisyncRcvFile("c:\temp\rcv.txt");
if RcvResult = 1 then
  LogMessage("receive no data condition occurred");
else if RcvResult = 2 then
  LogMessage("receive error condition occurred");
```

BisyncRcvMsg

Command format

```
integer BisyncRcvMsg();
```

Explanation of command

This command is used to receive data using the Bisync protocol. Data received is stored in a mailbox message with one attachment. This command can be used in conjunction with **SetRcvNoData** and **SetRcvError** to look for other data responses in place of or lack of a receive file.

Parameters

There are no parameters for this command.

Return value

0	Normal end of file.
1	If the **RcvNoData** condition occurred.
2	If the **RcvError** condition occurred.

Example

```
integer RcvResult;
SetRcvNoData("no data");
SetRcvError("error");
RcvResult = BisyncRcvMsg();
if RcvResult = 1 then
  LogMessage("receive no data condition occurred");
else if RcvResult = 2 then
  LogMessage("receive error condition occurred");
```

BisyncSetEof

Command format

```
BisyncSetEof(string EofChar);
```

Explanation of command

This command set the end of termination character for both sending and receiving.

Parameters

EofChar Contains the end of file character.

Return value

There are no return values for this command.

Example

```
BisyncSetEof("^26"); // set for ETB
```

BisyncSndAtm

Command format

```
BisyncSndAtm(integer MsgId, integer AtmId);
```

Explanation of command

This command is used to send a mailbox attachment using the Bisync protocol.

Parameters

MsgId	Must contain the mailbox message identifier returned from **MbxGetNextMsg.**
AtmId	Must contain the mailbox attachment identifier returned from **MbxGetNextAtm.**

Return value

There are no return values for this command.

Example

```
integer MsgId;
integer AtmId;
MbxStartMsgLoop();
while MbxGetNextMsg(MsgId) != 0 do
begin
   MbxStartAtmLoop(MsgId);
   while MbxGetNextAtm(AtmId) != 0 do
   begin
      BisyncSndAtm(MsgId, AtmId);
   end
end
```

BisyncSndCtl

Command format

```
BisyncSndCtl(string Data);
```

Explanation of command

This command transmits the specified string using Bisync protocol.

Parameters

Data	Contains the control string to be transmitted.

Return value

There are no return values for this command.

Example

```
BisyncSndCtl("hello world");
```

BisyncSndCtlEtx

Command format

```
BisyncSndCtlEtx(string Data);
```

Explanation of command

This command transmits the specified string using Bisync protocol and ends each block with an end of text (ETX).

Parameters

Data	Contains the control string to be transmitted.

Return value

There are no return values for this command.

Example

```
BisyncSndCtlEtx("hello world");
```

BisyncSndFile

Command format

```
BisyncSndFile(string FileSpec);
```

Explanation of command

This command is used to send a file using the Bisync protocol.

Parameters

FileSpec	A fully qualified filespec pointing to a file. UNC naming can be used.

Return value

There are no return values for this command.

Example

```
BisyncSndFile("c:\temp\snd.txt");
```

BisyncTable

Command format

```
BisyncTable(string Table);
```

Explanation of command

This command allows you to load new translation tables. The new tables must be in the same format as the asciiebc.ovr and ebcascii.ovr files.

Parameters

Table	Contains the file names separated by a space character.

Return value

There are no return values for this command.

Example

```
BisyncTable("newtbl.a2e newtbl.e2a");
```

Break

Command format

```
break;
```

Explanation of command

The **break** keyword terminates the execution of the nearest enclosing **while** loop, and passes control to the statement that follows the **end** keyword. The **break** keyword is generally used in complex loops to terminate a loop before several statements have been executed.

Parameters

There are no parameters for this command.

Return value

There are no return values for this command.

Example

```
while i<10 do
begin
if (i = 8) then
continue;
if (i = 9) then
break;
end
//As long as "i" has a value less than "10" the loop will repeat. If "i"
has
//a value of "8", the loop will continue. If "i" has a value
//of "9" the loop will terminate.
```

concat

Command format

```
concat(string1,string2,num_char);
```

Explanation of command

The concat function concatenates a specified number of characters from one string onto the end of another string.

Parameters

string1	The variable name of the first string of characters.
string2	The variable name of the second string of characters.
num_char	The number of characters from the second string to concatenate onto the end of the first string.

Return values

There are no return values for this command.

Example

```
string[10] s1,s2;
concat(s1,s2,5);
//Concatenate five characters from string "s2"
//onto the end of string "s1"
```

Continue

Command format

```
continue;
```

Explanation of command

The **continue** keyword continues the execution of the innermost loop without processing the statements in the loop that follow the **continue** statement.

Parameters

There are no parameters for this command.

Return value

There are no return values for this command.

Example

```
while i<10 do
begin
if (i = 8) then
continue;
if (i = 9) then
break;
end
//As long as "i" has a value of "8" the innermost
//loop will repeat. If "i" does not have a value of
//"8" the next statement (break statement)will be
//processed.
```

DclLogoff

Command format	`DclLogoff();`
Explanation of command	This command performs a logoff sequence using the DCL protocol.
Parameters	There are no parameters for this command.
Return value	There are no return values for this command.

DclLogon

Command format

```
DclLogon(string LogonAcct, string LogonId, string LogonPsw, string
IEAcct, string IEId, string IEPsw, string NewLogonPsw, string
NewIEPsw);
```

Explanation of command

This command performs a logon sequence using the DCL protocol.

Parameters

LogonAcct	Contains the Advantis logon account.
LogonId	Contains the Advantis logon identifier.
LogonPsw	Contains the Advantis logon password.
IEAcct	Contains the Information Exchange account.
IEId	Contains the Information Exchange identifier.
IEPsw	Contains the Information Exchange password.
NewLogonPsw	Contains the new Advantis logon password. This is only used if you want to change your logon password.
NewIEPsw	Contains the new Information Exchange password. This is only used if you want to change your Information Exchange password.

Return value

There are no return values for this command.

DclRcvMsg

Command format

```
DclRcvMsg([string UserMsgClass]);
```

Explanation of command

This command is used to receive data using the DCL protocol. Data received is stored in a mailbox message with one attachment.

Parameters

UserMsgClass	Optional. If specified, will retrieve data that contains the specified user message classification.

Return value

There are no return values for this command.

DclSetAck

Command format

```
DclSetAck(string AckType);
```

Explanation of command

This command is used to set the type of acknowledgment messages you want to receive from Information Exchange.

Parameters

AckType	Type of acknowledgment message you want to receive. Valid values are: R – Receipt acknowledgment D – Delivery acknowledgment B – Both receipt and delivery

Return value

There are no return values for this command.

Example

```
DclSetAck("R");
```

DclSndAtm

Command format

```
AsciiSndAtm(integer MsgId, integer AtmId);
```

Explanation of command

This command is used to send a mailbox attachment using the DCL protocol.

Parameters

MsgId	Must contain the mailbox message identifier returned from **MbxGetNextMsg.**
AtmId	Must contain the mailbox attachment identifier returned from **MbxGetNextAtm.**

Return value

There are no return values for this command.

Example

```
integer MsgId;
integer AtmId;
MbxStartMsgLoop();
while MbxGetNextMsg(MsgId) != 0 do
begin
   MbxStartAtmLoop(MsgId);
   while MbxGetNextAtm(AtmId) != 0 do
   begin
      DclSndAtm(MsgId, AtmId);
   end
end
```

DoRcv

Command format

```
integer DoRcv
```

Explanation of command

This command can be used to determine if a receive type session was requested (i.e. receive-only or send-receive). Since there is only one script per mailbox definition, this command provides an indication of the type of session that was requested so that the appropriate commands can be issued.

Parameters

There are no parameters for this command.

Return value

0	If the session is not a receive only, or send and receive type session.
1	If the session is a receive only, or send and receive type session

Example

```
if DoRcv then
begin
   // do something
end
```

DoSnd

Command format

```
integer DoSnd
```

Explanation of command

This command can be used to determine if a send type session was requested (i.e.send-only or send-receive). Since there is only one script per mailbox definition, this command provides an indication of the type of session that was requested so that the appropriate commands can be issued.

Parameters

There are no parameters for this command.

Return value

0	If the session is not a send only, or send and receive type session.
1	If the session is a send only, or send and receive type session.

Example

```
if DoSnd then
begin
   // do something
end
```

EiconCall

Command format	`EiconCall();`
Explanation of command	This command is used to make an X.25 connection using Eicon Technology hardware.
Parameters	There are no parameters for this command.
Return values	There are no return values for this command.

EiconListen

Command format	`EiconListen();`
Explanation of command	This command is used to listen for X.25 connections while using Advanced Data Distribution using Eicon Technology hardware.
Parameters	There are no parameters for this command.
Return values	There are no return values for this command.

EiconSetBICUG

Command format

```
EiconSetBICUG(integer Value1, integer Value2);
```

Explanation of command

This command is used to set the X.25 facilities for bilateral closed user group selection. See the ITU (International Telecommunications Union) X.25 recommendation (1984) for a more in-depth description and value settings.

Parameters

Value1	1st and 2nd digit index number to the bilateral closed user group.
Value2	3rd and 4th digit index number to the bilateral closed user group.

Return values

There are no return values for this command.

EiconSetClassNeg

Command format

```
EiconSetClassNeg(integer Value);
```

Explanation of command

This command is used to set the X.25 facilities throughput class negotiation. See the ITU (International Telecommunications Union) X.25 recommendation (1984) for a more in-depth description and value settings.

Parameters

Value	Throughput class values.

Return values

There are no return values for this command.

Example

```
EiconSetClassNeg(119);
// sets throughput class negotiation to 1200
```

EiconSetCUG

Command format

```
EiconSetCUG(integer Value);
```

Explanation of command

This command is used to set the X.25 facilities closed user group selection. See the ITU (International Telecommunications Union) X.25 recommendation (1984) for a more in-depth description and value settings.

Parameters

Value	index number to the closed user group.

Return values

There are no return values for this command.

EiconSetCUGOA

Command format

```
EiconSetCUGOA(integer Value);
```

Explanation of command

This command is used to set the X.25 facilities closed user group with outbound access selection. See the ITU (International Telecommunications Union) X.25 recommendation (1984) for a more in-depth description and value settings.

Parameters

Value	index number to the outgoing closed user group.

Return values

There are no return values for this command.

EiconSetNUI

Command format

```
EiconSetNUI(integer Length, string NUI);
```

Explanation of command

This command is used to set the X.25 facilities for network user identification. See the ITU (International Telecommunications Union) X.25 recommendation (1984) for a more in-depth description and value settings.

Parameters

Length	The length of the NUI string.
NUI	The contents of the NUI with a maximum password length of 6 bytes and a maximum length of 8 bytes.

Return values

There are no return values for this command.

EiconSetPacketSize

Command format

```
EiconSetPacketSize(integer Remote, integer Local);
```

Explanation of command

This command is used to set the X.25 facilities packet size. See the ITU (International Telecommunications Union) X.25 recommendation (1984) for a more in-depth description and value settings.

Parameters

Remote	Remote packet size code.
Local	Local packet size code.

Return values

There are no return values for this command.

Example

```
EiconSetPacketSize(7,7);
// sets remote and local window size to 128 bytes
```

EiconSetRevFast

Command format

```
EiconSetRevFast(integer Value);
```

Explanation of command

This command is used to set the X.25 facilities for reverse charging, or fast select, or both. See the ITU (International Telecommunications Union) X.25 recommendation (1984) for a more in-depth description and value settings.

Parameters

Value	The value for reverse charging, or fast select, or both.

Return values

There are no return values for this command.

EiconSetUserData

Command format

```
EiconSetUserData(integer P1, integer P2, integer P3, integer P4, string
UserData)
```

Explanation of command

This command is used to set the X.29 call user data parameters.

Parameters

P1	Protocol ID 1
P2	Protocol ID2
P3	Protocol ID 3
P4	Protocol ID 4
UserData	User-defined data (maximum of 12 bytes).

Return values

There are no return values for this command.

Example

```
EiconSetUserData(192,0,0,0,"")
//sets all 4 protocol Ids and does not include user data

EiconSetUserData(192,0,0,0,"user data")
//sets all 4 protocol Ids and includes user data
```

EiconSetWindowSize

Command format

```
EiconSetPacketSize(integer Remote, integer Local);
```

Explanation of command

This command is used to set the X.25 facilities window size. See the ITU (International Telecommunications Union) X.25 recommendation (1984) for a more in-depth description and value settings.

Parameters

Remote	Remote window size.
Local	Local window size.

Return values

There are no return values for this command.

Example

```
EiconSetWindowSize(7,7);
// sets remote and local window size to 7
```

FtpCD

Command format

```
FtpCD(string Directory);
```

Explanation of command

This command is used to change directories on the FTP server by using a CD command instead of the normal CWD command. This is used by GEIS.

Parameters

Directory	The directory to change to on the FTP server.

Return value

There are no return values for this command.

FtpChangeDir

Command format

```
FtpChangeDir(string Directory);
```

Explanation of command

This command is used to change directories on the FTP server. The FtpChangeDir will issue a CWD command first, and if that fails and a 502 error is received, it will retry using the CD command.

Parameters

Directory	The directory to change to on the FTP server.

Return value

There are no return values for this command.

FtpDelete

Command format

```
FtpDelete(string FileName);
```

Explanation of command

This command is used to delete a file on the FTP server.

Parameters

FileName	The filename to delete on the FTP server.

Return value

There are no return values for this command.

FtpDoCmd

Command format

```
integer FtpDoCmd(string Cmd, [string Arg]);
```

Explanation of command

This command is used to issue RFC compliant FTP commands.

Parameters

Cmd	The RFC compliant FTP command to issue.
Arg	The FTP command arguments. Optional

Return value

Returns the value found at the beginning of the last line returned from the host, or a value greater than 500 if an error occurs. It is the user's responsibility to evaluate this code to determine if the command was successful.

Example

```
integer Result;
string[10] sResult;
string[20] sFinal;

Result = FtpDoCmd("RMD", "test");
ntoa(Result, sResult);
sFinal = "DoCmd=" + sResult;
logMessage(sFinal);
```

FtpGetDir

Command format

```
integer FtpGetDir(integer bUseList);
```

Explanation of command

This command is used to obtain a directory listing from the FTP server. This command is used in conjunction with FtpGetDirString to inspect each line of the directory output returned by the server.

Parameters

bUseList	A boolean variable indicating the directory listing type. If the value is TRUE, a LIST command is sent to the server. If the value is FALSE, a NLST command is sent to the server.

Return value

The integer value returned indicates how many directory rows/lines were returned by the server.

Example

```
integer iCnt;
integer iIndex;
string[100] OneLine;

iCnt = FtpGetDir(LIST);  // TRUE=LIST  FALSE=NLST
iIndex = 0;
while iIndex < iCnt do
begin
   FtpGetDirString(iIndex, OneLine);
   // do something to find exact filename in the line
   FtpRcvMsg(OneLine);
   iIndex = iIndex + 1;
end
```

FtpGetDirString

Command format

```
FtpGetDirString(integer iIndex, string buffer);
```

Explanation of command

This command is used to return each line of a directory output returned by the FTP server. A call to FtpGetDir must occur before this function is used, so that the directory output can be received.

Parameters

iIndex	The zero-based index of the directory output array.
buffer	A string buffer to place the contents of that directory line, based on the array index.

Return value

This command does not return a value.

Example

```
integer iCnt;
integer iIndex;
string[100] OneLine;

iCnt = FtpGetDir(LIST);  // TRUE=LIST  FALSE=NLST
iIndex = 0;
while iIndex < iCnt do
begin
   FtpGetDirString(iIndex, OneLine);
   // do something to find exact filename in the line
   FtpRcvMsg(OneLine);
   iIndex = iIndex + 1;
end
```

FtpHost

Command format

```
FtpHost();
```

Explanation of command

This command is used to start a FTP server session.

Parameters

This command does not have any parameters.

Return value

This command does not return a value.

Example

```
FtpHost();
SetStatus(SUCCESS);
```

FtpRcvFile

Command format

```
integer FtpRcvFile(string LocalFile, string RemoteFile, integer
DeleteAfterRcv);
```

Explanation of command

This command is used to receive a file using the FTP protocol. This command can be used in conjunction with **SetRcvNoData** and **SetRcvError** to look for other responses in place of, or lack of, a file.

Parameters

LocalFile	The name of the file to create on the local system.
RemoteFile	The name of the file on the remote system.
DeleteAfterRcv	Optional. Defaults to 0 or FALSE which does not delete thefiles. Set to 1 or TRUE to delete each file after the receivecompletes.

Return value

0	If the receive completed successfully.
1	If the RcvNoData condition occurred.
2	If the RcvError condition occurred.

Example

```
FtpRcvFile("localfile.txt", "remotefile.txt", TRUE);
```

FtpRcvMsg

Command format

```
FtpRcvMsg(string filename, integer DeleteAfterRcv);
```

Explanation of command

This command is used to receive one file using the FTP protocol. Data received is stored in a mailbox message with one attachment.

Parameters

FileName	The filename to receive from the FTP server.
DeleteAfterRcv	Optional. Defaults to 0 or FALSE which does not delete thefiles. Set to 1 or TRUE to delete each file after the receivecompletes.

Return value

There are no return values for this command.

Example

```
FtpRcvMsg("filename.txt", TRUE);
```

FtpRcvMsgAll

Command format

```
FtpRcvMsgAll([integer DeleteAfterRcv]);
```

Explanation of command

This command is used to receive all files from the current directory on an FTP server. Data received is stored in a mailbox message with one attachment.

Parameters

DeleteAfterRcv	Optional. Defaults to 0 or FALSE which does not delete thefiles. Set to 1 or TRUE to delete each file after the receivecompletes.

Return value

There are no return values for this command.

Example

```
FtpRcvMsgAll(TRUE);
```

FtpRename

Command format

```
FtpRename(string OldName, string NewName);
```

Explanation of command

This command is used to rename a file on an FTP server.

Parameters

OldName	Specifies the name of the existing file to be renamed.
NewName	Specifies the new file name.

Return value

There are no return values for this command.

Example

```
FtpRename("oldfile", "newfile");
```

FtpSetMode

Command format

```
FtpSetMode(integer Mode);
```

Explanation of command

This command is used to set the mode in which files are transferred to the FTP server.

Parameters

Mode	Specify either ASCII mode or BINARY mode. The default is BINARY mode.

Return value

There are no return values for this command.

Example

```
FtpSetMode(ASCII);
```

FtpSndAtm

Command format

```
FtpSndAtm(integer MsgId, integer AtmId,
string FileName);
```

Explanation of command

This command is used to send a mailbox attachment using the FTP protocol.

Parameters

MsgId	Must contain the mailbox message identifier returned from **MbxGetNextMsg**.
AtmId	Must contain the mailbox attachment identifier returned from **MbxGetNextAtm**.
FileName	Contains the name of the file to create on the FTP server.

Return value

There are no return values for this command.

Example

```
integer MsgId;
integer AtmId;
string[128] FileName;
MbxStartMsgLoop();
while MbxGetNextMsg(MsgId) != 0 do
begin
   MbxStartAtmLoop(MsgId);
   while MbxGetNextAtm(AtmId) != 0 do
   begin
      MbxGetAtmFileName(MsgId, AtmId, FileName);
      FtpSndAtm(MsgId, AtmId, FileName);
   end
end
```

FTPSndFile

Command format

```
FtpSndFile (string LocalFilename, string RemoteFilename,
integer DeleteFlag);
```

Explanation of command

This command is used to send a file that's not in the mailbox system to an FTP server using either the FTP transport or the WSFTP transport.

Parameters

LocalFilename	This variable specifies the local filename of the file to send.
RemoteFilename	This variable specifies the remote filename on the FTP server that should be created when the file is sent.
DeleteFlag	Specifies whether the local file should be deleted after successfully sending the file. Valid Values are: TRUE (delete the local file after successfully sending the file) or FALSE (do not delete the local file after successfully sending.). This variable must be specified as either TRUE or FALSE. There is no default.

Return value

There are no return values for this command.

get

Command format

```
integer_variable = get datetime_component (datetime_variable);
```

Explanation of command

The **get** function enables you to access individual components of a datetime variable.

Parameters

datetime_componentThe individual component of the datetime variable.

datetime_variableThe datetime variable of which you want to access a component part.

Return values

This function returns the integer value of the datetime_component.

Example

```
integer a;
integer b;
datetime d;
a = get days (d);
b = get hours (d);
//Accesses the days from the datetime variable "d"
//and loads into variable "a". Accesses the hours
//from the datetime variable "d" and loads into
//variable "b".
```

GetSessionTry

Command format

```
integer GetSessionTry();
```

Explanation of command

This command is used to determine which session attempt, initial or retry is currently running.

Return values

The return value is “0” if it is the initial session attempt. The return value is greater than “0” for each session retry.

Example

```
if GetSessionTry() = 0 then
   LogMessage("initial try");
else if GetSessionTry() = 1 then
   LogMessage("1st retry");
else if GetSessionTry() = 2 then
   LogMessage("2nd retry");
```

If…Then…Else

Command format

```
if condition then
```

Explanation of command

The **if**, **then**, and **else** keywords allow the use of conditional logic in scripts. Gentran:Server uses conditional logic to test conditions and then, depending on the results of the test, perform operations. Conditions can be nested to any level. The condition is typically a comparison, but it can be any expression that concludes with a numeric value. Gentran:Server interprets the value as either *true* or *false*. The system interprets a zero value as false and a nonzero value as true.

Gentran:Server evaluates the **if…then** condition, and if it is true, the system runs all the statements that follow the **then** keyword. If the condition is false, none of the statements following **then** are run.

You can use the **else** keyword in conjunction with **if…then** to define several blocks of statements, one of which will be executed. Gentran:Server tests the first **if…then** condition. If the condition is false, the system proceeds to test each sequential condition until it finds one that is true. The system runs the corresponding block of statements for the true condition. If none of the **if…then** conditions are true, the system runs the statements following the **else** keyword.

Parameters

There are no parameters for this command.

Return values

There are no return values for this command.

Example

```
if condition then
 begin
   statement1;
   statement2;
 end
else if condition then
 begin
   statement3;
   statement4;
 end
```

KermitRcvMsg

Command format

```
KermitRcvMsg();
```

Explanation of command

This command is used to receive data using the Kermit protocol.

Parameters

There are no parameters for this command.

Return value

There are no return values for this command.

Example

```
if DoRcv then
begin
   AsciiSndCtl("D");
   AsciiRcvCtl("Your choice");
   AsciiSndCtl("Z");
   AsciiRcvCtl("File name?");
   AsciiSndCtl("*.*^0D");
   AsciiRcvCtl("Begin your transfer procedure...");
   KermitRcvMsg();
end
```

KermitSndAll

Command format

```
KermitSndAll();
```

Explanation of command

This command is used to send all files using the Kermit protocol.

Parameters

There are no parameters for this command.

Return value

There are no return values for this command.

Example

```
if DoSnd then
begin
   AsciiSndCtl("U");
   AsciiRcvCtl("Your choice");
   AsciiSndCtl("Z");
   AsciiRcvCtl("File name?");
   AsciiSndCtl("*^0D");
   AsciiRcvCtl("Begin your transfer procedure...");
   KermitSndAll();
end
```

KermitSet8thBitQuote

Command format

```
KermitSet8thBitQuote(integer 8thBitQuote);
```

Explanation of command

Specifies the character (in decimal notation) to be used as the 8th bit quoting character. Use this command to override the default 8thBitQuote value.

Parameters

8thBitQuote	Decimal representation of the character to be used as the 8th bit quoting character. Must be in the range (33 – 126). The default is 0 for not using 8th bit quoting.

Return value

There are no return values for this command.

KermitSetBlockCheckType

Command format

```
KermitSetBlockCheckType(integer BlockCheckType);
```

Explanation of command

Specifies the type of block check to use. Use this command to override the default BlockCheckType value.

Parameters

BlockCheckType 1 = 1 byte checksum.
2 = 2 byte checksum.
3 = 3 byte CRC.
The default is 1.

Return value

There are no return values for this command.

KermitSetBlockStart

Command format

```
KermitSetBlockStart(integer BlockStart);
```

Explanation of command

Specifies the character (in decimal notation) to be used as the block start. Use this command to override the default BlockStart value.

Parameters

BlockStart	Decimal representation of the character to be used as the block start. Must be in the range (0 – 127). The default is 1 (SOH).

Return value

There are no return values for this command.

KermitSetControlQuote

Command format

```
KermitSetControlQuote(integer ControlQuote);
```

Explanation of command

Specifies the character (in decimal notation) to be used as the control quote. Use this command to override the default ControlQuote value.

Parameters

ControlQuote	Decimal representation of the character to be used as the control quote. Must be in the range of (32 – 127). The default is 35 (#).

Return value

There are no return values for this command.

KermitSetEndOfLine

Command format

```
KermitSetEndOfLine(integer EndOfLine);
```

Explanation of command

Specifies the character (in decimal notation) to be used as the end of line. Use this command to override the default EndOfLine value.

Parameters

EndOfLine	Decimal representation of the character to be used as the end of line. Must be in the range (0 – 127). The default is 13 (CR).

Return value

There are no return values for this command.

KermitSetNumberOfPads

Command format

```
KermitSetNumberOfPads(integer NumberOfPads);
```

Explanation of command

Specifies the number of pad characters to use. Use this command to override the default NumberOfPads value.

Parameters

NumberOfPads	Number of pad characters to use. Must be in the range (0 – 127). The default is 0 for none.

Return value

There are no return values for this command.

KermitSetPacketSize

Command format

```
KermitSetPacketSize(integer PacketSize);
```

Explanation of command

Specifies the maximum packet size to be used during file transfer. Use this command to override the default PacketSize value.

Parameters

PacketSize	Contains the maximum size of a Kermit data packet during file transfer. The default is 1024.

Return value

There are no return values for this command.

KermitSetPadCharacter

Command format

```
KermitSetPadCharacter(integer PadCharacter);
```

Explanation of command

Specifies the character (in decimal notation) to be used as the pad character. Use this command to override the default PadCharacter value.

Parameters

PadCharacter	Decimal representation of the character to be used as the pad character. Must be in the range (0 – 127). The default is 0 (NULL).

Return value

There are no return values for this command.

KermitSetParity

Command format

```
KermitSetParity(integer Parity);
```

Explanation of command

Specifies that parity is in use and should be accounted for. Use this command to override the default Parity value.

Parameters

Parity	0 (zero) or 1. Specify 1 to inform the Kermit protocol that parity is in use and requires special processing. The default is 0 for no parity.

Return value

There are no return values for this command.

left

Command format

```
string_variable = left(string_variable,num_char);
```

Explanation of command

The **left** function extracts a specified number of character from the left side of a string variable or field and returns the result as a string.

Parameters

string_variable	A variable defined as type string.
num_char	The number of characters to count from the left side of a string.

Return values

This function returns the string value of the variable.

Example

```
string [25]name;
string [5]temp_variable;
name = "Acme Shipping Company"
temp_variable = left(name,4);
// "temp_variable" would contain "Acme"
```

len

Command format

```
number_char = len(string);
```

Explanation of command

The **len** function is a numerical function that counts and returns the number of characters in a string. The numerical functions enable you to convert one data type to another.

Parameters

string	The string you wish to evaluate.

Return values

This function returns the length of a string.

Example

```
integer a;
a = len("hello");
// "a" contains the value 5
```

LogMessage

Command format

```
LogMessage(string UserMsg);
```

Explanation of command

This command is used to write a user specified string or message to the session log.

Parameters

UserMsg	A string value to be written to the session log.

Return value

There are no return values for this command.

Example

```
integer RcvResult
SetRcvNoData("no data");
SetRcvError("error");
RcvResult = AsciiRcvMsg("^04");
if RcvResult = 1 then
  LogMessage("receive no data condition occurred");
else if RcvResult = 2 then
  LogMessage("receive error condition occurred");
```

MbxGetAtmContentType

Command format

```
MbxGetAtmContentType(integer AtmId, string ContentType, string
ContentSubType);
```

Explanation of command

This command is used to retrieve the content type and content subtype from the attachment.

Parameters

AtmId	Must contain the attachment identifier for the attachment to be queried.
ContentType	A string value used to return the content type.
ContentSubType	A string value used to return the content subtype.

Return value

There are no return values for this command.

Example

```
integer AtmId;
integer Length1;
integer Length2;
string[100] SndCmd;
string[20]  ContentType;
string[20]  ContentSubType;
string[8]   UserClass;
SndCmd = "+SEND";
MbxGetAtmContentType(AtmId, ContentType, ContentSubType);  // format is
MsgClass_x[8] or DataFormat_x[8]
Length1 = len(ContentType);
Length2 = len(ContentSubType);
if strstr(ContentType, "MsgClass_") != -1 then
   UserClass = mid(ContentType, 9, Length1-9);
else if strstr(ContentSubType, "MsgClass_") != -1 then
   UserClass = mid(ContentSubType, 9, Length2-9);
SndCmd = SndCmd + " USERCLASS(" + UserClass + ")";
```

MbxGetAtmFileExt

Command format

```
MbxGetAtmFileExt(integer MsgId, integer AtmId, string FileExt);
```

Explanation of command

This command is used to obtain the message attachment file extension. If the attachment contains C:\TEST\TEXT.TXT, this function returns TXT.

Parameters

MsgId	Must contain the mailbox message identifier returned from **MbxGetNextMsg.**
AtmId	Must contain the mailbox attachment identifier returned from **MbxGetNextAtm.**
FileExt	A string value that will be used to return the attachment file extension.

Return value

There are no return values for this command.

Example

```
integer MsgId;
integer AtmId;
string[20] FileExtension;
MbxStartMsgLoop();
while MbxGetNextMsg(MsgId) != 0 do
begin
   MbxStartAtmLoop(MsgId);
   while MbxGetNextAtm(AtmId) != 0 do
   begin
      MbxGetAtmFileExt(MsgId, AtmId, FileExt);
      // do something
   end
end
```

MbxGetAtmFileName

Command format

```
MbxGetAtmFileName(integer MsgId, integer AtmId, string Filename);
```

Explanation of command

This command is used to obtain the message attachment filename. If the attachment contains C:\TEST\TEXT.TXT, this function returns TEXT.TXT.

Parameters

MsgId	Must contain the mailbox message identifier returned from **MbxGetNextMsg.**
AtmId	Must contain the mailbox attachment identifier returned from **MbxGetNextAtm.**
Filename	A string value that will be used to return the attachment filename.

Return value

There are no return values for this command.

Example

```
integer MsgId;
integer AtmId;
string[128] FileName;
MbxStartMsgLoop();
while MbxGetNextMsg(MsgId) != 0 do
begin
   MbxStartAtmLoop(MsgId);
   while MbxGetNextAtm(AtmId) != 0 do
   begin
      MbxGetAtmFileName(MsgId, AtmId, FileName);
      // do something
   end
end
```

MbxGetAtmFilePath

Command format

```
MbxGetAtmFilePath(integer MsgId, integer AtmId, string FilePath);
```

Explanation of command

This command is used to obtain the message attachment file path. If the attachment contains C:\TEST\TEXT.TXT, this function returns C:\TEST\TEXT.TXT.

Parameters

MsgId	Must contain the mailbox message identifier returned from **MbxGetNextMsg.**
AtmId	Must contain the mailbox attachment identifier returned from **MbxGetNextAtm.**
FilePath	A string value that will be used to return the attachment file path.

Return value

There are no return values for this command.

Example

```
integer MsgId;
integer AtmId;
string[128] FilePath;
MbxStartMsgLoop();
while MbxGetNextMsg(MsgId) != 0 do
begin
   MbxStartAtmLoop(MsgId);
   while MbxGetNextAtm(AtmId) != 0 do
   begin
      MbxGetAtmFilePath(MsgId, AtmId, FilePath);
      // do something
   end
end
```

MbxGetAtmFileTitle

Command format

```
MbxGetAtmFileTitle(integer MsgId, integer AtmId, string FileTitle);
```

Explanation of command

This command is used to obtain the message attachment file title. If the attachment contains C:\TEST\TEXT.TXT, this function returns TEXT.

Parameters

MsgId	Must contain the mailbox message identifier returned from **MbxGetNextMsg.**
AtmId	Must contain the mailbox attachment identifier returned from **MbxGetNextAtm.**
FileTitle	A string value that will be used to return the attachment file title.

Return value

There are no return values for this command.

Example

```
integer MsgId;
integer AtmId;
string[128] FileTitle;
MbxStartMsgLoop();
while MbxGetNextMsg(MsgId) != 0 do
begin
   MbxStartAtmLoop(MsgId);
   while MbxGetNextAtm(AtmId) != 0 do
   begin
      MbxGetAtmFileTitle(MsgId, AtmId, FileTitle);
      // do something
   end
end
```

MbxGetNextAtm

Command format

```
integer MbxGetNextAtm(integer AtmId)
```

Explanation of command

This command returns the next available mailbox attachment identifier that it ready to be sent.

Parameters

AtmId	This variable is used as the Return value.

Return value

Non zero	If a valid mailbox attachment identifier is available and ready to be sent.
zero	If no more mailbox attachments are available to send.

Example

```
integer MsgId;
integer AtmId;
MbxStartMsgLoop();
while MbxGetNextMsg(MsgId) != 0 do
begin
   MbxStartAtmLoop(MsgId);
   while MbxGetNextAtm(AtmId) != 0 do
   begin
      // do something
   end
end
```

MbxGetNextMsg

Command format

```
integer MbxGetNextMsg(integer MsgId)
```

Explanation of command

This command returns the next available mailbox message identifier that is ready to be sent.

Parameters

MsgId	This variable is used as the Return value.

Return value

Non zero	if a valid mailbox message identifier is available and ready to be sent.
zero	if no more mailbox messages are available to send.

Example

```
integer MsgId;
MbxStartMsgLoop();
while MbxGetNextMsg(MsgId) != 0 do
begin
   // do something
end
```

MbxGetOriginalMsgId

Command format

```
MbxGetOriginalMsgId(integer MsgId, string OriginalMsgId);
```

Explanation of command

This command is used to retrieve the original message identifier from the message.

Parameters

MsgId	The mailbox message identifier for the message to be queried
OriginalMsgId	A string value used to return the information.

Return value

There are no return values for this command.

Example

```
integer MsgId;
integer Length;
string[100] SndCmd;
string[10]  OriginalMsgId;
string[8]   MsgName;
string[5]   MsgSeqn;
SndCmd = "+SEND";
// code for reconciliation
MbxGetOriginalMsgId(MsgId, OriginalMsgId);
Length = len(OriginalMsgId);
if Length > 8 then
begin
   MsgName = left(OriginalMsgId, 8);
   MsgSeqn = mid(OriginalMsgId, 8, Length-8);
   SndCmd = SndCmd + " MSGNAME(" + MsgName + ") MSGSEQN(" + MsgSeqn +
")";
end
else
begin
   MsgName = OriginalMsgId;
   SndCmd = SndCmd + " MSGNAME(" + MsgName + ")";
end
```

MbxGetRcvrEmailAddr

Command format

```
MbxGetRcvrEmailAddr(integer MsgId, string RcvrEmailAddr);
```

Explanation of command

This command is used to retrieve the receiver gateway email address from the message.

Parameters

MsgId	The mailbox message identifier for the message to be queried.
RcvrEmailAddr	A string value used to return the receiver's gateway email address.

Return value

There are no return values for this command.

Example

```
integer MsgId;
integer Slash;
integer Length;
integer Underscore;
string[100] SndCmd;
string[80]  RcvrEmailAddr;
string[8]   DestAcct;
string[8]   DestUserId;
string[3]   SysId;
SndCmd = "+SEND";
MbxGetRcvrEmailAddr(MsgId, RcvrEmailAddr);
// format is DESTACCT[8]_DESTUID[8]/SYSID[3] or LIST=X[8]
Length = len(RcvrEmailAddr);
if strstr(RcvrEmailAddr, "LIST=") >= 0 then
begin
   DestAcct = mid(RcvrEmailAddr, 5, Length-5);
   SndCmd = SndCmd + " LIST(" + DestAcct + ")";
end
else
begin
   Underscore = strstr(RcvrEmailAddr, "_");
   DestAcct = left(RcvrEmailAddr, Underscore);
   Underscore = Underscore + 1;
   Slash = strstr(RcvrEmailAddr, "/");
   // look for SYSID
   if ( Slash >= 0 ) then
   begin
      DestUserId = mid(RcvrEmailAddr, Underscore, Slash-Underscore);
      Slash = Slash + 1;
      SysId = mid(RcvrEmailAddr, Slash, Length-Slash);
      SndCmd = SndCmd + " SYSID(" + SysId + ")";
   end
   else
      DestUserId = mid(RcvrEmailAddr, Underscore, Length-Underscore);
      SndCmd = SndCmd + " DESTACCT(" + DestAcct + ") DESTUID(" +
DestUserId + ")";
   end
end
```

MbxLogon

Command format

```
integer MbxLogon(string mailbox, string password);
```

Explanation of message

This command is used in Advanced Data Distribution to collect the mailbox ID and Advanced Data Distribution password.

Parameters

mailbox	The Advanced Data Distribution mailbox name.
password	The Host password.

Return value

There are no return values for this command.

Example

```
// Sample Pool Host Script
string[50] Mbx;
string[50] Psw;
// look for LOGON MBX=x PSW=x
AsciiRcvCtl("^0D");
if RcvBufSearch("LOGON") = 0 then
begin
   AsciiSndCtl("GOODBYE^0D");
   Exit();
end
// validate logon
ParseStart();
ParseNamed("MBX=", " ^0D", Mbx);
ParseNamed("PSW=", " ^0D", Psw);
if MbxLogon(Mbx, Psw) = 0 then
begin
   AsciiSndCtl("GOODBYE^0D");
   Exit();
end
```

MbxStartAtmLoop

Command format

```
MbxStartAtmLoop(integer MsgId);
```

Explanation of command

This command is used in conjunction with **MbxStartMsgLoop**, **MbxGetNextMsg**, and **MbxGetNextAtm** to iterate through all available mailbox attachments that are ready to send for a particular message. This command must precede **MbxGetNextAtm** and can only be used after establishing a message loop.

Parameters

MsgId	Must contain the mailbox message identifier returned from **MbxGetNextMsg**.

Return value

There are no return values for this command.

Example

```
integer MsgId;
integer AtmId;
MbxStartMsgLoop();
while MbxGetNextMsg(MsgId) != 0 do
begin
   MbxStartAtmLoop(MsgId);
   while MbxGetNextAtm(AtmId) != 0 do
   begin
      // do something
   end
end
```

MbxStartMsgLoop

Command format

```
MbxStartMsgLoop();
```

Explanation of command

This command is used in conjunction with **MbxGetNextMsg** to iterate through all available mailbox messages that are ready to send. This command must precede **MbxGetNextMsg**.

Parameters

There are no parameters for this command.

Return value

There are no return values for this command.

Example

```
integer MsgId;
MbxStartMsgLoop();
while MbxGetNextMsg(MsgId) != 0 do
begin
   // do something
end
```

mid

Command format

```
string_variable = mid(string_variable,start_pos,num_char);
```

Explanation of command

The **mid** function extracts from a specified position in a string, either to the end of the string or for a specified number of characters and returns the resultant string.

Parameters

string_variable	The variable containing the string you want to extract.
start_pos	The starting position in the string of characters.
num_char	The number of characters from the starting position.

Return values

This function returns the value of the substring identified by the parameters.

Example

```
string [25]name;
string [10]temp_variable;
name = "Acme Shipping Company"
temp_variable = mid(name,5,8);
// "temp_variable" would contain "Shipping"
```

ntoa

Command format

```
string = ntoa(real,string);
```

Explanation of command

The **ntoa** function is a numerical function that converts real numbers into strings. The numerical functions enable you to convert one data type to another.

Parameters

real	The number or variable you wish to convert.
string	The name of the string you wish to store the converted number in.

Return values

There are no return values for this command.

Example

```
real b;
string[8] s;
b = 5.5;
ntoa(5.5, s);
//The variable "s" contains the string "5.5".
```

OftpAddSfidCheck

Command format

```
OftpAddSfidCheck(string CheckOriginator);
```

Explanation of command

This command adds the specified string to a table of valid originator IDs that will be checked against the sfidorig field of the SFID when receiving files. The file is rejected if the contents of the sfidorig field do not match any of the values in the table. This command must be issued for every ID that needs to be in the table and must proceed the **OftpRemote** or **OftpHost** script command.

Parameters

`CheckOriginator`	String value of the originator ID to be checked against the SFID sfidorig field.

Return value

There are no return values for this command.

Example

This is an example of the command.

```
OftpAddSfidCheck("ID1");
OftpAddSfidCheck("ID2");
OftpRemote(OftpId, OftpPsw, OftpNewPsw);
```

Related topic

See the "Working with OFTP" appendix in the *Communications Gateway Configuration Guide* or in the *Advanced Data Distribution Configuration Guide* for more information about using the OFTP protocol.

OftpHost

Command format

```
OftpHost(string SSIDcode, string SSIDpwsd, string SSIDuser,
[integer LogonUsingSSID]);
```

Explanation of command

This command is used to perform Advanced Data Distribution functions when a trading partner initiates a communications session to an OFTP server. This command will take care of all sending and/or receiving, depending upon what type of session was requested, without specifying any of the mailbox-type commands.

Parameters

SSIDcode	A string value that will be sent in the ssidcode field of the SSID.
SSIDpwsd	A string value that will be sent in the ssidpwsd field of the SSID.
SSIDuser	A string value that will be sent in the ssiduser field of the SSID.
LogonUsingSSID	Optional value. If set to TRUE, will use the incoming remote SSID information to log on to mailbox instead of the MbxLogon script command.

Return value

There are no return values for this command.

Example 1

This is an example of an OftpHost command.

```
AsciiSndCtl("IODETTE FTP READY ^0D");
OftpHost("SAMPLE ODETTE FTP HOST", "OFTP PSW", "");
SetStatus(SUCCESS);
```

Example 2

This is an example of an OftpHost command containing the optional LogonUsingSSID value.

```
AsciiSndCtl("IODETTE FTP READY ^0D");
OftpHost("SAMPLE ODETTE FTP HOST", "OFTP PSW", "", TRUE);
SetStatus(SUCCESS);
```

OftpNoEerps

Command format

```
OftpNoEerps();
```

Explanation of command

This command instructs the system not to send any EERPs for files received and not to expect any EERPs for files sent.

Parameters

There are no parameters for this command.

Return value

There are no return values for this command.

Related topic

See the “Working with OFTP” appendix in the *Communications Gateway Configuration Guide* or in the *Advanced Data Distribution Configuration Guide* for more information about using the OFTP protocol.

OftpRemote

Command format

```
OftpRemote(string OftpId, string OftpPsw,
string OftpNewPsw, [string IdToCheck, string PswToCheck]);
```

Explanation of command

This command is used to perform a complete remote session to an OFTP server. This command will take care of all sending and/or receiving, depending upon what type of session was requested, without specifying any of the mailbox-type commands. The optional parameters are used to verify that the ID and password information contained in the SSID is accurate. If a discrepancy is found, the session fails. The optional parameters must be used in conjunction with each other.

Parameters

OftpId	Contains the OFTP user ID.
OftpPsw	Contains the OFTP user password.
OftpNewPsw	Contains the OFTP new user password. This is only used for certain systems that allow changing of the password. If used, it will be placed in the ssiduser field of the ssid structure.
IdToCheck	Verifies that the OftpId contained in the SSID is accurate. Must be used in conjunction with PswToCheck.
PswToCheck	Verifies that the OftpPsw contained in the SSID is accurate. Must be used in conjunction with IdToCheck.

Return value

There are no return values for this command.

Related topic

See the “Working with OFTP” appendix in the *Communications Gateway Configuration Guide* or in the *Advanced Data Distribution Configuration Guide* for more information about using the OFTP protocol.

OftpSetDynamicPassword

Command format

```
OftpSetDynamicPassword([integer Switch], [integer PswExpire]);
```

Explanation of command

This command is used to dynamically update the OFTP password on both the local machine and the host machine, provided the host supports password changes. In order for this command to work properly, scriptvar-type variables must be used for the first three parameters of the OftpRemote command. There are three forms of this command which are shown in the examples below.

Parameters

Switch	Optional. Integer value that identifies which field of the SSID to use for the new OFTP password. If set to FALSE, which is the default, the new password is sent in the ssiduser field of the SSID and the old password is sent in the ssidpwsd field. If set to TRUE, the new password is sent in the ssidpwsd field of the SSID and the old password is sent in the ssiduser field.
PswExpire	Optional. Integer value that identifies (in days) when the password expires. If this parameter is used, the Switch parameter must also be specified. When the password expires, a new password is generated automatically on behalf of the user.

Return values

There are no return values for this command.

Example

```
scriptvar string[25] OftpId;
scriptvar string[8] OftpPsw;
scriptvar string[8] OftpNewPsw;
AsciiRcvCtl("IODETTE FTP READY ^0D");

// Example1, OftpNewPsw is sent in the ssiduser field.
OftpSetDynamicPassword();
// same as OftpSetDynamicPassword(FALSE);

// Example2, OftpNewPsw is sent in the ssidpswd field.
OftpSetDynamicPassword(TRUE);

// Example3, after 30 days, a new password is generated
// automatically and sent in the ssiduser field.
OftpSetDynamicPassword(FALSE, 30);

// Once the session completes, OftpPsw is updated with
// the new password and OftpNewPsw is set to null.
OftpRemote(OftpId, OftpPsw, OftpNewPsw);
```

Related topic

See the "Working with OFTP" appendix in the *Communications Gateway Configuration Guide* or in the *Advanced Data Distribution Configuration Guide* for more information about using the OFTP protocol.

OftpSetEerpDelivered

Command format

```
OftpSetEerpDelivered();
```

Explanation of command

An EERP that is received for a file sent is, by default, marked as PICKEDUP in the mailbox system. The OftpSetEerpDelivered()command overrides the default setting and marks the EERP as DELIVERED.

Parameters

There are no parameters for this command.

Return value

There are no return values for this command.

Example

```
OftpSetEerpDelivered();
OftpRemote(OftpID, OftpPsw, OftpNewPsw);
```

Related topic

See the "Working with OFTP" appendix in the *Communications Gateway Configuration Guide* or in the *Advanced Data Distribution Configuration Guide* for more information about using the OFTP protocol.

OftpSetMaxRecordSize

Command format

```
OftpSetMaxRecordSize(integer RecordSize);
```

Explanation of command

This command overrides the default record size of 0 (used with the default record format of "U"). This command is often used in conjunction with OftpSetRecordFormat.

Parameters

RecordSize specifies the record size used with record formatting.

Return value

There are no return values for this command.

Example

```
OftpSetMaxRecordSize(80);  // sets 80 byte record size
OftpSetRecordFormat("F");    // sets fixed record formatting
```

Related Topic

See *OftpSetRecordFormat* on page 2 - 110 for more information.

OftpSetRcvDupCheck

Command format

```
OftpSetRcvDupCheck();
```

Explanation of command

When files are received during OftpRemote or OftpHost sessions, this command checks for and rejects duplicate files based on the sfiddsn, sfiddate, sfidtime, and sfidorig field values of the SFID.

Parameters

There are no parameters for this command.

Return value

There are no return values for this command.

Related topic

See the “Working with OFTP” appendix in the *Communications Gateway Configuration Guide* or in the *Advanced Data Distribution Configuration Guide* for more information about using the OFTP protocol.

OftpSetRecordFormat

Command format

```
OftpSetRecordFormat(string RecordFormat);
```

Explanation of command

This command overrides the default record format "U," which is used for the entire communications session. If a record format override is specified in the content type of the message attachment, the system overrides the OftpSetRecordFormat command. OftpSetRecordFormat is often used in conjunction with **OftpSetMaxRecordSize**.

Parameters

RecordFormat Valid values are "F," "V," "U," and "T."

Return value

There are no return values for this command.

Example

```
OftpSetRecordFormat("F");     // sets fixed record formatting
OftpSetMaxRecordSize(80);  // sets 80 byte record size
```

Related Topic

See *OftpSetMaxRecordSize* on page 2 - 108 for additional information.

OftpSetSpecialEERP

Command format

```
OftpSetSpecialEERP();
```

Explanation of command

This command specifies use of special end-to-end response packet processing where the destination and originator fields are swapped. This is normally used for TRADANET type networks.

Parameters

There are no parameters for this command.

Return value

There are no return values for this command.

Related topic

See the "Working with OFTP" appendix in the *Communications Gateway Configuration Guide* or in the *Advanced Data Distribution Configuration Guide* for more information about using the OFTP protocol.

OftpSpecialLogicOff

Command format

```
OftpSpecialLogicOff();
```

Explanation of command

This command is used to turn off the initial special logic setting used during negotiation. It applies only when using an X.25 connection and must be used **before** the X.25 connection is made. The remote site may still negotiate to use special logic.

Parameters

There are no parameters for this command.

Return value

There are no return values for this command.

Related topic

See the "Working with OFTP" appendix in the *Communications Gateway Configuration Guide* or in the *Advanced Data Distribution Configuration Guide* for more information about using the OFTP protocol.

ParseNamed

Command format

```
ParseNamed(string sName, string sEndChars, string sChunk);
```

Explanation of command

This command extracts a “chunk” from the buffer by looking for an identifying “name” in the data. It searches the whole buffer, regardless of the current parsing position. This allows the script to handle named parameters that can occur in any order. After extracting a chunk, the current parsing position will be advanced if the chunk was beyond the previous parsing position. This means that if you call **ParseNext** after calling **ParseNamed** several times, **ParseNext** will begin its search from the end of the named “chunk” that was furthest into the string.

Parameters

sName	A string value to match in the data.
sEndChars	A string of characters used to identify the end point of the search.
sChunk	A string value to retrieve from the data.

Return value

There are no return values for this command.

Example

```
// Sample Script
string[50] Mbx;
string[50] Psw;
// look for LOGON MBX=x PSW=x
AsciiRcvCtl("^0D");
if RcvBufSearch("LOGON") = 0 then
begin
   AsciiSndCtl("GOODBYE^0D");
   Exit();
end
// validate logon
ParseStart();
ParseNamed("MBX=", " ^0D", Mbx);
ParseNamed("PSW=", " ^0D", Psw);
if MbxLogon(Mbx, Psw) = 0 then
begin
   AsciiSndCtl("GOODBYE^0D");
   Exit();
end
```

ParseNext

Command format

```
ParseNext(string sSkipChars, string sEndChars, string sChunk);
```

Explanation of command

This command extracts the next "chunk" of data from the buffer. It begins at the current parsing position and finds the start of the chunk by skipping over characters until it encounters a character that is **not** identified in the sSkipChars parameter. This command then locates the end of the chunk once it finds a character listed in the sEndChars parameter, and sets the current parsing position accordingly.

Parameters

sSkipChars	A string of characters to ignore in the buffer.
sEndChars	A string of characters used to identify the end point of the search.
sChunk	A string value to retrieve from the data

Return values

There are no return values for this command.

Example

```
// Sample Script
string[50] Command;
string[50] Mbx;
string[50] Psw;
// Reset the parser
ParseStart();
// Get the command, it is always at the start of
// the buffer
ParseNext("", "", Command);
ParseNamed("MBX=", " ^0D", Mbx);
ParseNamed("PSW=", " ^0D", Psw);
if MbxLogon(Mbx, Psw) = 0 then
begin
   AsciiSndCtl("GOODBYE^0D");
   Exit();
end
```

ParseSkip

Command format

```
ParseSkip(number nSkip);
```

Explanation of command

This command advances the current parse position by nSkip characters.

Parameters

nSkip	The number of characters to skip.

Return values

There are no return values for this command.

ParseStart

Command format

```
ParseStart();
```

Explanation of command

This command must be called prior to parsing each buffer from the remote computer. It resets the current parsing position to the start of the buffer.

Parameters

There are no parameters for this command.

Return values

There are no return values for this command.

Example

```
// Reset the parser and validate logon
ParseStart();
ParseNamed("MBX=", " ^0D", Mbx);
ParseNamed("PSW=", " ^0D", Psw);
if MbxLogon(Mbx, Psw) = 0 then
begin
   AsciiSndCtl("GOODBYE^0D");
   Exit();
end
```

Pause

Command format	`Pause(integer Seconds);`
Explanation of command	This command is used to pause execution of the script for the specified number of seconds.
Parameters	Seconds Contains the number of seconds to pause.
Return value	There are no return values for this command.

RcvBufSearch

Command format

```
integer RcvBufSearch(string SearchStr)
```

Explanation of message

This command is used to search the last receive buffer for the specified string. This command is useful for receiving some control information and then determining what was received.

Parameters

SearchStr	Contains the string to search for in last receive buffer.

Return value

0	If the string buffer does not contain the search string.
1	If the string buffer does contain the search string.

Example

```
AsciiRcvCtl("^0D");
if RcvBufSearch("hello" ) = 1 then
   LogMessage("found hello");
```

right

Command format

```
string_variable = right(string_variable,num_char);
```

Explanation of command

The **right** function extracts a specified number of characters from the right side of a string variable or field.

Parameters

string_variable	The name of the string of characters you wish to manipulate.
num_char	The number of characters to count from the right side of a string.

Return values

This function returns the characters from the string.

Example

```
string [25]name;
string [10]temp_variable;
name = "Acme Shipping Company"
temp_variable = right(name,7);
// "temp_variable" would contain "Company"
```

scriptvar

Command format

```
scriptvar [optional] type name;
```

Explanation of command

This command defines a script variable that can be used for editing in the user interface. Script variables are not required, but when used, they provide a user-friendly interface to the user for capturing specific information. By assigning a meaningful name to a script variable, users can readily determine what information needs to be provided. When you create a script variable, users are required to enter a value for that variable unless you have specified that the variable is optional by using the "optional" parameter.

Parameters

optional	Identifies this script variable as an optional variable.
type	Type of variable (integer, real, string, datetime or array).
name	The name of the variable.

Return value

There are no return values for this command.

Example

```
scriptvar string[10] MailboxId;
scriptvar optional integer SomeNumber;
```

set

Command format

```
set datetime_component (datetime_variable,integer_variable);
```

Explanation of command

The **set** function enables you to define individual components of a datetime variable.

Parameters

datetime_componentThe individual component of the datetime variable.

datetime_variableThe datetime variable of which you want to access a component part.

integer_variable An integer variable.

Return values

There are no return values for this command.

Example

```
integer a;
integer b;
datetime d;
set days (d,a);
set hours (d,a);
//Defines the days of the datetime variable "d" from
//variable "a".
//Defines the hours of the datetime variable "d" from
//variable "b".
```

SetBlockSize

Command format

```
SetBlockSize(integer BlockSize);
```

Explanation of command

This command is used to set or reset the blocksize used for various protocols.

Parameters

BlockSize	Contains the value to set or reset the blocksize used during transfers.

Return value

There are no return values for this command.

SetBufferSize

Command format	`SetBufferSize(integer BufferSize);`
Explanation of command	This command is used to reset the size of the internal read/write buffers. The default is 2048.
Parameters	BufferSize Contains the new size of the internal read/write buffers.
Return value	There are no return values for this command.

SetRcvError

Command format

```
SetRcvError(string RcvErrorIndicator);
```

Explanation of command

This command is used to specify a string to scan for during downloads using certain protocols to indicate an error condition. This is useful in situations where you might receive data or just a string indicating an error occurred.

Parameters

RcvErrorIndicatorA string value that indicates a receive error condition.

Return value

There are no return values for this command.

SetRcvFileMode

Command format

```
SetRcvFileMode(integer Mode);
```

Explanation of command

This command controls whether the system overwrites, or appends to, files created locally.

Parameters

Mode	valid values are APPEND or OVERWRITE.

Return value

There are no return values for this command.

Example

```
SetRcvFileMode(APPEND);
SetRcvFileMode(OVERWRITE);
```

SetRcvNoData

Command format

```
SetRcvNoData(string RcvNoDataIndicator);
```

Explanation of command

This command is used to specify a string to scan for during downloads using certain protocols to indicate no data is available to receive. This is useful in situations where you might receive data or just a string indicating no data is available to receive.

Parameters

RcvNoDataIndicator A string value that indicates a receive no data condition.

Return value

There are no return values for this command.

SetSessionType

Command format

```
SetSessionType(integer Type);
```

Explanation of command

This command overrides the type of communications session the system performs. Use this command if the system cannot start communications with a particular session type.

Parameters

Type	Value values are:	
	0	= Send Only
	1	= Send/Rcv
	2	= Rcv Only

Return value

There are no return values for this command.

Example

```
SetSessionType(0);
```

SetSpecialUpdate

Command format

```
SetSpecialUpdate(integer TrueOrFalse);
```

Explanation

This command overrides the communications session's normal operations. When this command is enabled, any calls to SndOk in the script are overridden and will not be marked sent until the SetStatus(SUCCESS) command is executed. Also, any data received is not sent to the mailbox system until the SetStatus(SUCCESS) command is executed. This command is useful when communicating to certain networks that use "session level," otherwise known as "all or nothing" logic.

Parameters

TrueOrFalse Boolean flag used to enable or disable this command.

Return Value

There are no return values for this command.

SetStatus

Command format

```
SetStatus( integer Status );
```

Explanation of command

This command sets the session status. Use one of the predefined constant values either SUCCESS or FAILED.

Parameters

Status	Contains the value to set the session status (either SUCCESS or FAILED).

Return value

There are no return values for this command.

Example

```
SetStatus(SUCCESS);
```

SetTimeout

Command format

```
SetTimeout(integer Seconds);
```

Explanation of command

This command sets a new timeout value used during communications sessions.

Parameters

Seconds new timeout value in seconds (default 60).

Return value

There are no return values for this command.

Example

```
SetTimeout(60);
```

SndOK

Command format

```
SndOK(integer MsgId)
SndOK(integer MsgId, integer AtmId)
```

Explanation of command

This command is used to indicate a successful send of a mailbox message. This command must be issued once a message and all its attachments have been sent successfully so that the status indicators can be updated and so that the message will not be sent again. This command is also issued when an attachment has been sent successfully so that the status indicators can be updated and so that the attachment will not be sent again.

Parameters

MsgId	The mailbox message identifier to update the status for. The mailbox message identifier for the attachment specified in AtmId.
AtmId	The mailbox attachment identifier to update the status for.

Return value

There are no return values for this command.

Example

```
integer MsgId;
integer AtmId;
MbxStartMsgLoop();
while MbxGetNextMsg(MsgId) != 0 do
begin
   MbxStartAtmLoop(MsgId);
   while MbxGetNextAtm(AtmId) != 0 do
   begin
      AsciiSndAtm(MsgId, AtmId, "^04");
      SndOK(MsgId, AtmId);
   end
   SndOK(MsgId);
end
```

strdate

Command format

```
strdate(datetime,"format",string);
```

Explanation of command

The **strdate** function converts a datetime type into a string using a format that you specify. This function allows you to include static characters such as a slash (/), which gives you access to full date support.

Parameters

datetime — The datetime variable (month specified as 0-11).

format — The desired date format. The format specifiers are as follows:

Format Specifier	Description
%a	Abbreviated weekday name
%A	Full weekday name
%b	Abbreviated month name
%B	Full month name
%d	Day of the month as a decimal number (01 – 31)
%H	Hour in 24-hour format (00 – 23)
%I	Hour in 12-hour format (01– 12)
%j	Day of the year as a decimal number (001 – 366)
%m	Month as a decimal number (01 – 12)
%M	Minute as a decimal number (00 – 59)
%S	Second as a decimal number (00 – 59)
%U	Week of the year as a decimal number, with Sunday as the first day of the week (00 – 51)
%w	Weekday as a decimal number (0 – 6, with Sunday as "0")

(Continued on next page)

(Contd) Format Specifier	Description
%W	Week of the year as a decimal number, with Monday as the first day of the week (00 – 51)
%y	Year without the century as a decimal number (00 – 99)
%Y	Year with the century as a decimal number
%%	Percent sign

string The string variable.

Return values

There are no return values for this command.

Example

```
datetime d;
string[8] s;

strdate(d,"%y/%m%d",s);
//Converts a datetime variable into an eight
//character string in the format "year/month/day".
```

strstr

Command format

```
integer = strstr("string","substring");
```

Explanation of command

The **strstr** function finds a substring inside a string.

Parameters

integer	An integer variable.
string	The string to evaluate.
substring	The part of the string you are interested in.

Return values

This function returns the character position of the first instance of the designated substring within the specified string, or -1 if the substring is not found.

Example

```
integer d;
d = strstr("mississippi","is");

//Finds the first instance of the substring "is"
//inside the string "mississippi" and returns the
//position of that first substring.
```

TipRemote

Command format

```
TipRemote();
```

Explanation

This command is used to perform a complete remote session to a Tradanet Network using the Tradanet Interface Protocol (TIP). To use this command, the user must first enable Tradanet commands through the mailbox configuration properties, select TIP commands, and then configure the options accordingly.

Return Value

There are no return values for this command.

Example

```
TipRemote();
SetStatus(SUCCESS);
```

TipSetFwd

Command format

```
TipSetFwd(integer Forwarding);
```

Explanation

This command sets the block forwarding characteristics used during a TipRemote session.

Parameters

Forwarding 0 - no forwarding character present

1 - forwarding character present on input to service

2 - forwarding character present on output from service

3 - (default) forwarding character present on both input and output.

Return Value

There are no return values for this command.

Example

```
TipSetFwd(0);
TipRemote();
SetStatus(SUCCESS);
```

TipSetPadding

Command format

```
TipSetPadding(string Padding);
```

Explanation

This command sets the block padding characteristics used during a TipRemote session.

Parameters

Padding	N - (default) No padding. All blocks use physical size
	Y - Pad data blocks. Pad last block of data to blocksize.
	A - Pad all blocks (directives and data) to blocksize.

Return Value

There are no return values for this command.

Example

```
TipSetPadding("A");
TipRemote();
SetStatus(SUCCESS);
```

TipSetPadChar

Command format

```
TipSetPadChar(integer PadChar);
```

Explanation

This command sets the pad character used for block padding during a TipRemote session.

Parameters

PadChar Default 0. Character used to pad blocks to blocksize.

Return Value

There are no return values for this command.

Example

```
TipSetPadChar(32);     // set to blank/space
TipRemote();
SetStatus(SUCCESS);
```

TipSetConType

Command format

```
TipSetConType(integer ConType);
```

Explanation

This command sets the connection type used during a TipRemote session.

Parameters

ConType	1 - LAN
	2 - Default. X25
	3 - Other

Return Value

There are no return values for this command.

Example

```
TipSetConType(3);
TipRemote();
SetStatus(SUCCESS);
```

While…Do

Command format

```
while condition do
```

Explanation of command

The **while...do** keyword runs a statement repeatedly until the specified termination condition equals zero. The system tests the terminating condition *before* each iteration of the loop, so a **while** loop executes zero or more times depending on the value of the termination expression.

Parameters

There are no parameters for this command.

Return value

There are no return values for this command.

Example

```
integer i;

while i < 10 do
begin
if (i = 8) then
continue;
if (i = 9) then
break;
end
//While "i" is less than ten, execute the loop. If "i" is equal to or
greater
//than ten, terminate the loop.
```

winexec

Command format

```
integer_variable = Winexec("Program",window_display)
```

Explanation of command

The **winexec** function enables you to execute another program while running the translator. This program is executed asynchronously. You specify the program and determine how you want the program window displayed. You can also return an error code, if desired. If the error code is greater than 32, the program ran without errors. If the error code is less than 32, the program did not run because of an error. If the error code is "0," the system is out of memory. If the error code is "2," you didn't specify a file name. The error code is *not* the return value from the program you executed.

Parameters

program An executable program name string.

window_display A number that indicates how you want the program window displayed. This table lists the window_display numbers that control the appearance of the program window. Use the *number* to indicate how you want the program window displayed, not the window_display value.

Number	Window_Display	Definition
0	SW_HIDE	Hides the window and activates another window.
1	SW_SHOWNORMAL	Activates and displays a window. If the window is minimized or maximized, it is restored to the original size and position. This flag should be specified when displaying a window for the first time.
1	SW_NORMAL	Activates and displays a window in the original size and position.
2	SW_SHOWMINIMIZED	Activates the window and displays it as a minimized windows.
3	SW_SHOWMAXIMIZED	Activates the window and displays it as a maximized window.
3	SW_MAXIMIZE	Maximizes the window.
4	SW_SHOWNOACTIVATE	Displays the window in its most recent size and position, but does not activate it (the current active window remains active).

(Continued on next page)

(Contd) Number	Window_Display	Definition
5	SW_SHOW	Activates the window and displays it in its current size and position.
6	SW_MINIMIZE	Minimizes the window.
7	SW_SHOWMINNOACTIVE	Displays the window as a minimized window without activating it (the current active window remains active).
8	SW_SHOWNA	Displays the window in its most recent size and position without activating it (the current active window remains active).
9	SW_RESTORE	Activates and displays the window. If the window was minimized or maximized, it is restored to its original size and position.
10	SW_SHOWDEFAULT	Activates the window and allows Windows to determine the size and position.

Return value

This function returns the result value provided by the operating system.

Example

```
winexec("program.exe", 3)

//Executes the "program.exe" program asynchronously.
//The program window is displayed maximized (3).
```

WwaLogoff

Command format	`WwaLogoff();`
Explanation of command	This command performs a logoff sequence using the World Wide Async (WWA) protocol.
Parameters	There are no parameters for this command.
Return value	There are no return values for this command.

WwaLogon

Command format

```
WwaLogon(string LogonAcct, string LogonId, string LogonPsw, string
IEAcct, string IEId, string IEPsw, string NewLogonPsw, string
NewIEPsw);
```

Explanation of command

This command performs a logon sequence using the World Wide Async protocol.

Parameters

LogonAcct	Contains the logon account.
LogonId	Contains the logon identifier.
LogonPsw	Contains the logon password.
IEAcct	Contains the Information Exchange account.
IEId	Contains the Information Exchange identifier.
IEPsw	Contains the Information Exchange password.
NewLogonPsw	Contains the new logon password. This is only used if you want to change your logon password.
NewIEPsw	Contains the new Information Exchange password. This is only used if you want to change your Information Exchange password.

Return value

There are no return values for this command.

WwaRcvMsg

Command format

```
WwaRcvMsg([string UserMsgClass]);
```

Explanation of command

This command is used to receive data using the World Wide Async protocol. Data received is stored in a mailbox message with one attachment.

Parameters

UserMsgClass	Optional. If specified, will retrieve data that contains the specified user message classification.

Return value

There are no return values for this command.

WwaSndAtm

Command format

```
WwaSndAtm(integer MsgId, integer AtmId);
```

Explanation of command

This command is used to send a mailbox attachment using the World Wide Async protocol.

Parameters

MsgId	Must contain the mailbox message identifier returned from **MbxGetNextMsg.**
AtmId	Must contain the mailbox attachment identifier returned from **MbxGetNextAtm.**

Return value

There are no return values for this command.

Example

```
integer MsgId;
integer AtmId;
MbxStartMsgLoop();
while MbxGetNextMsg(MsgId) != 0 do
begin
   MbxStartAtmLoop(MsgId);
   while MbxGetNextAtm(AtmId) != 0 do
   begin
      WwaSndAtm(MsgId, AtmId);
   end
end
```

XmodemRcvFile

Command format

```
XmodemRcvFile(string Filename);
```

Explanation of command

This command is used to receive a file using the Xmodem protocol. The command SetRcvFileMode controls whether the system overwrites, or appends to, the file. The default is for the system to overwrite the file.

Parameters

Filename	The name of the file that is used for receiving data. Universal Naming Convention (UNC) names are valid.

Return value

There are no return values for this command.

Example

```
SetRcvFileMode(APPEND);
XmodemRcvFile("testfile.txt");
```

XmodemRcvMsg

Command format

```
XmodemRcvMsg();
```

Explanation of command

This command is used to receive data using the Xmodem protocol. Data received is stored in a mailbox message with one attachment.

Parameters

There are no parameters for this command.

Return values

There are no return values for this command.

XmodemSetFillChar

Command format

```
XmodemSetFillChar(string FillChar);
```

Explanation of command

This command sets the fill character used in the Xmodem protocol to pad the last block of data if the data does not fill the entire block. The default fill character is 0 (NULL).

Parameters

FillChar	Specifies the character to use to fill the end of the last data block, if needed. Can be a hex value in the form of ^xx as shown in the example below.

Return value

There are no return values for this command.

Example

```
XmodemSetFillChar("^1A");
```

XmodemSndAll

Command format

```
XmodemSndAll();
```

Explanation of command

This command is used to send all available message attachments as one file using the Xmodem protocol. Because the command internally accomplishes some tasks, there is no need to:

- define a loop for each message and attachment
- use the SndOk script command.

Parameters

There are no parameters for this command.

Return value

There are no return values for this command.

Example

```
If DoSnd
   XmodemSndAll();
```

XmodemSndAtm

Command format

```
XmodemSndAtm(integer MsgId, integer AtmId);
```

Explanation of command

This command is used to send a mailbox attachment using the Xmodem protocol.

Parameters

MsgId	Must contain the mailbox message identifier returned from **MbxGetNextMsg.**
AtmId	Must contain the mailbox attachment identifier returned from **MbxGetNextAtm.**

Return value

There are no return values for this command.

Example

```
integer MsgId;
integer AtmId;
MbxStartMsgLoop();
while MbxGetNextMsg(MsgId) != 0 do
begin
   MbxStartAtmLoop(MsgId);
   while MbxGetNextAtm(AtmId) != 0 do
   begin
      XmodemSndAtm(MsgId, AtmId);
   end
end
```

XmodemSndFile

Command format

```
XmodemSndFile(string Filename);
```

Explanation of command

This command is used to send a file using the Xmodem protocol.

Parameters

Filename	The name of the file to be sent. UNC names can be used.

Return value

There are no return values for this command.

Example

```
XmodemSndFile("testfile.txt");
```

ZmodemRcvFile

Command format

```
ZmodemRcvFile(string FilenameOrPath [, integer PathIndicator]);
```

Explanation of command

This command instructs the system on how to receive a file or multiple files that use the Zmodem protocol. When a file is received, by default it overwrites a pre-existing file.

When used in conjunction with the SetRcvFileMode command, you can instruct the system to append the information received to a pre-existing file.

Parameters

FilenameOrPath	Specifies either a fixed filename or a path. If using a fixed filename, all files received are written to this single file. If specifying a path name, a separate file is created in that location for each file received. Universal Naming Convention (UNC) names are valid entries.
PathIndicator	Optional. The default setting is FALSE. If set to TRUE, you must specify a path name parameter.

Return value

There are no return values for this command.

Example 1

```
SetRcvFileMode(APPEND);
ZmodemRcvFile("testfile.txt");  // receives into a single file called
testfile.txt
```

Example 2

```
ZmodemRcvFile"("c:\temp\", TRUE);  // receives multiple files into the
temp directory
```

ZmodemRcvMsg

Command format

```
ZmodemRcvMsg();
```

Explanation of command

This command is used to receive data using the Zmodem protocol.

Parameters

There are no parameters for this command.

Return value

There are no return values for this command.

Example

```
if DoRcv then
begin
   AsciiSndCtl("D");
   AsciiRcvCtl("Your choice");
   AsciiSndCtl("Z");
   AsciiRcvCtl("File name?");
   AsciiSndCtl("*.*^0D");
   AsciiRcvCtl("Begin your transfer procedure");
   ZmodemRcvMsg();
end
```

ZmodemSndAll

Command format

```
ZmodemSndAll();
```

Explanation of command

This command is used to send all files using the Zmodem protocol.

Parameters

There are no parameters for this command.

Return value

There are no return values for this command.

Example

```
if DoSnd then
begin
   AsciiSndCtl("U");
   AsciiRcvCtl("Your choice");
   AsciiSndCtl("Z");
   AsciiRcvCtl("File name?");
   AsciiSndCtl("*^0D");
   AsciiRcvCtl("Begin your transfer procedure");
   ZmodemSndAll();
end
```

ZmodemSndFile

Command format

```
ZmodemSndFile(string Filename);
```

Explanation of command

This command is used to send a file using the Zmodem protocol.

Parameters

Filename	The name of the file to be sent. UNC names can be used.

Return value

There are no return values for this command.

Example

```
ZmodemSndFile("testfile.txt");
```