Gentran:Server® for Windows®

スクリプト言語 リファレンス ガイド

バージョン 5.0

Sterling Commerce
An IBM Company

4232-500-USER06-0000

著作権に関する表示

スターリング コマース ソフトウェア
営業秘密に関する表示

GENTRAN:SERVER FOR WINDOWS ソフトウェア (「スターリング コマース ソフトウェア」) は、米国スターリング コマース社、その関連企業またはそのライセンサの機密と営業秘密を有する財産であり、プロダクト使用契約の条件に基づいて提供されます。事前の書面による許可のない複製または開示は禁じられています。権利は制限されています。

本資料、本資料で述べられたスターリング コマース ソフトウェア、およびそれらに含まれる情報とノウハウは、スターリング コマース、その関連企業またはそのライセンサの専有的な、秘密の、財産的価値を有する営業秘密であり、許可されていない目的のために使用することおよび適正なスターリング コマースからの事前の書面による承認を受けることなく第三者に開示することは禁じられています。本資料と本資料で述べられたスターリング コマース ソフトウェアは、その複製、変更および使用を禁止または規制する条項が規定されたプロダクト使用契約に基づいて提供されます。複製を許可された場合には、複製が部分的であるか全体的であるかを問わず、その複製物にこの営業秘密表示とスターリングコマースの著作権表示を表示するものとします。

FAR に規定されるところにより米国の政府機関、政府の委託業者又はさらにその者の委託業者に対して本資料が提供された場合は、Title 48 CFR 52.227-19 による制限付きの権利として提供されます。また、DFAR に規定されるところにより米国の政府機関、政府の委託業者又はさらにその者の委託業者に対して本資料とスターリング コマース ソフトウェアが提供された場合は、商業用ソフトウェアと商業用ソフトウェア資料に関する Title 48 CFR 227-7202 の記載に基づくスターリング コマースの一般的な実施許諾契約に基づいて提供されます。

これらの条件の準拠法は、米国オハイオ州法 （法の抵触に関する規定を除く）が適用されます。締結済みの契約に基づいてスターリング コマース ソフトウェアを使用している場合には、これらの条件は当該締結済み契約に優先するものではなく、また、これを修正するものでもありません。

本資料に記載されている製品名は、該当各社の商標または登録商標である場合があります。Gentran 及び Gentran:Server はスターリングコマースの登録商標です。

サード パーティ ソフトウェア
スターリング コマース ソフトウェアの一部には、サード パーティ (「サード パーティ ライセンサ」) から提供された製品 (「サード パーティ ソフトウェア」) が含まれる場合や、サード パーティ ソフトウェアと同一の記録媒体で配布される場合があります。

保証の放棄
本資料と本資料で述べられたスターリング コマース ソフトウェアは「現状のもの」として、またはスターリング コマースのプロダクト使用契約で規定された「限定保証」とともに提供されるものとします。「限定保証」以外には、商品性および特定目的への適合性を含むいかなる明示的および黙示的保証も行われないものとします。スターリングコマースは適宜本表示を改訂し、又はその内容を変更できるものとし、その改訂又は変更をいかなる個人又は法人に対しても通知する義務を負わないものとします。

サード パーティ ソフトウェアは、商品性の黙示的保証および特定目的への適合性を含めて (ただしこれらに限定されない)、いかなる保証ならびに明示的および黙示的保証も伴わずに「現状のもの」として提供され、いかなる保証も行われません。また、米国国内に居住するか、本ソフトウェアを米国国内で使用している場合、所有権または権利の非侵害に関する明示的および黙示的保証は行われません。

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

目次

本書について

スクリプト言語の概要

スクリプト言語リファレンス

本書について

目次

はじめに

概要

『スクリプト言語 リファレンス ガイド』では、Gentran:Server for Windows コミュニケーションズ サブシステム用に付属しているスクリプト言語について説明します。

本書の対象読者

このマニュアルは、次の読者を対象としています。

- Gentran:Server システム管理者
- Gentran:Server for Windows の上級ユーザー

必要な知識

このソフトウェアを使用する読者は、次について習熟している必要があります。

- Microsoft® Windows
- Gentran:Server for Windows
- コミュニケーションズ プロトコル

本書の内容

はじめに

このセクションでは、本書の構成を説明します。

章の構成

このマニュアルの章構成は次のとおりです。各章の概要を説明します。

- 「本書について」では、このマニュアルの内容および構成について説明します。
- 「スクリプト言語の概要」では、スクリプト言語コマンドの概要について説明します。
- 「スクリプト言語リファレンス」では、すべてのスクリプト言語コマンドをアルファベット順に示します。

オンライン ヘルプ

はじめに

本書の内容の大半が、コミュニケーションズ ゲートウェイと、アドバンスド データ ディストリビューションのオンライン ヘルプ システムにも記載されています。

サポート情報

はじめに

スターリング コマース社では、Gentran:Server ソフトウェアに関する質問や問題についてお答えするため、熟練した製品サポート担当者によるサポートを提供しています。

メモ
Gentran:Server のカスタマ サポートは、スターリング コマース社以外の製品 (SQL Server 、Oracle など) のサポートは行っていません。ただし、他社の製品を Gentran:Server と併用できるように構成するサポートは提供しています。

電話番号

ご利用になるサポート電話番号は、『インストール準備 カード』を参照してください。

サポートに連絡する前に

迅速なサポートを提供するため、以下の事項についてご協力をお願いします。

- まず、発生した問題を再現してみて、イベントの正確な順序を記録してください。
- 製品サポートへのお問い合わせの際には、下記の情報をご用意ください。

情報	説明
ユーザー情報	会社名、氏名、電話番号および内線番号、およびケース番号 (以前に報告された問題を照会する場合)。
システム構成	Gentran:Server バージョン (およびインストールされているサービス パック)、プライマリ Gentran システム コントローラおよび問題が発生しているすべてのコンピュータの情報。Windows オペレーティング システムのバージョン、搭載メモリ、有効ディスク容量、データベース バージョン、Microsoft Data Access (MDAC) のバージョン、および Internet Explorer のバージョン。 ハードウェア、ソフトウェア、およびシステムの構成に対して加えた最近の変更も記入してください。
システム データ ストア	どのコンピュータがシステム データ ストア内にフォルダを格納しているのか。

(次のページへ続く)

情報	説明
エラー メッセージ	表示されたエラー メッセージの正確な語句表現とソフトウェアでのエラーが発生した時点、およびログ ファイルも記録してください。
試みた解決策	問題の解決を試みた際の手順とその結果の記録、推定される問題の発生回数と問題を再生することができるかどうかを報告してください。

スターリング コマース社 Support Web Site (英語) へのアクセス

スターリング コマース社 Customer Support Web Site は Gentran:Server for Windows のサポートに関する以下の重要な情報が記載されています。

- サポート サービスの範囲
- カスタマ サポート ポリシー
- 優先コール
- カスタマ サポートの電話番号一覧
- Support on Demand ケースの作成方法
- Support on Demand ケースのステータスの確認方法
- Support on Demand ケースへの情報追加の方法

Customer Support Web Site は常にアップデートされており、スターリング コマース製品のユーザーの皆様にご利用いただけます。この Web サイトには、最新の製品アップデート情報が記載されています。製品情報に関する重要な情報源としてご活用ください。

参照
Customer Support World Wide Web Site へのアクセス方法に関する情報については、『インストール準備 カード』を参照してください。

マニュアル

Customer Support Web Site にはドキュメント ライブラリがあり、Gentran:Server for Windows マニュアル セットがすべて含まれています。随時このライブラリから製品マニュアルを PDF フォーマットでダウンロードできます。

第 1 章

スクリプト言語の概要

目次

概要

本章の内容

この章では、スクリプト言語の概要を説明します。

コマンド参照ページの構成

各コマンドは、次の構成で説明されます。

- コマンド書式 — スクリプトで使うコマンドの書式を示します。
- コマンドの説明 — コマンドについて簡単に説明します。
- パラメータ — コマンドで使うパラメータがあれば、そのリストを示します。
- 戻り値 — 戻り値があれば、そのリストを示します。
- 例 — スクリプト内のコマンドの使い方を示す簡単な例です。

コマンドの書式

コマンドは次の書式を使って示されています。

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

各コマンドには、パラメータが後ろにいくつか付く場合や、付かない場合があります。パラメータは、丸かっこ () で囲み、コンマ (,) で区切ります。パラメータ名では、大文字と小文字が区別されます。コマンドにパラメータが伴わない場合は、空の丸かっこ付きになります。

各コマンドのパラメータには、そのデータ型が示されています。上記の例では、`Required_Parameter` が文字列 (string) パラメータとして定義され、1 つ以上の印刷可能文字を保持するために使われます。`Optional_Parameter` は整数 (integer) パラメータとして定義され、小数点以下を持たない自然数を保持するために使われます。

パラメータの種類

コマンドは、次のような 2 種類のパラメータを伴います。

- オプション パラメータ — 角かっこ ([ ]) 内に指定されます。
- 必須のパラメータ — 角かっこを付けずに指定されます。

パラメータの宣言

スクリプト コマンド内のすべてのパラメータは、スクリプトで使う前に宣言しなければなりません。宣言は、データ型とその型のパラメータから構成されます。

例

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

データ型

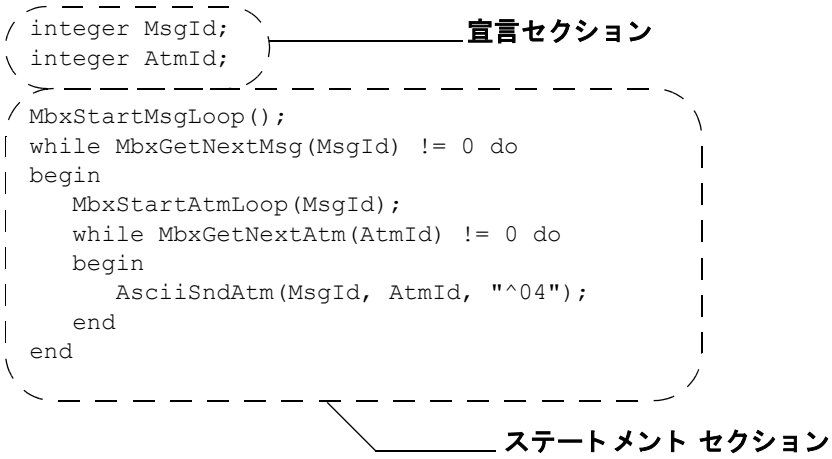
パラメータのデータ型は次のとおりです。

- 整数 (integer) — 小数部を持たない自然数です。
- 実数 (real) — 小数部を持つことができる数です。
- 文字列 (string) — 1 個以上の印刷可能文字が入ります。
- 日付 / 時刻 (datetime) — 日付または時刻です。
- 配列 (array) — 1 つのデータ型が繰り返し登場するテーブルです。

スクリプトの書式

スクリプトは、次の 2 つのセクションから構成されます。スクリプトで使うパラメータを宣言するための宣言セクションと、実際のスクリプト コマンドを指定するステートメント セクションの 2 つです。

例



キーワード

キーワードは、スクリプトのフローを制御するためにスクリプト内で使う特殊コマンドです。キーワードは、定義済みの値に一致するかどうかをテストするために条件付き論理で使います。

例

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

この例では、RcvResult の値が数値の 1 に等しいときにログ メッセージが生成されます。**if…then** キーワードでは、スクリプトを制御して、RcvResult の値が 1 でない場合にはログ メッセージが生成されないようにします。

利用できるキーワード

コミュニケーション スクリプトで使用可能なものとして、次のキーワードが用意されています。

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

式

式は、システムが評価する論理単位です。

例

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

演算子

演算子は、式内で最も単純な演算を定義します。次の表に、スクリプト コマンドで使う演算子を示します。

演算子	説明
+	加算、連結
-	減算
*	乗算
/	除算
=	代入、等価
>	より大
<	より小
>=	より大か等しい
<=	より小か等しい
!=	非等号
!	論理否定
&	論理積
|	論理和
<<	日付変更

行終了符号

宣言とステートメントは、セミコロン (;) で終了します。

例外
キーワード ステートメント (**if...then** や **while...do** など) は、セミコロンでは終了しません。セミコロンで終了するのは、これらのキーワード ステートメント内にあるステートメントのみです。

例

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

第 2 章

スクリプト言語リファレンス

目次

AnsiClearRcvMsg

コマンド書式

```
AnsiClearRcvMsg();
```

コマンドの説明

このコマンドは、AnsiClear プロトコルを使ってデータを受信するために使用します。受信データは、1 つの添付と一緒にメールボックス メッセージに格納されます。

パラメータ

このコマンドには、パラメータはありません。

戻り値

このコマンドには、戻り値はありません。

AnsiClearSetRcvRsp

コマンド書式

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

コマンドの説明

このコマンドは、データを受信したときに応答を送信するよう設定します。すべてのシステムで応答が要求されているわけではないため、既定では応答は送信されません。

パラメータ

Response	送信する応答が入ります。1 つ以上の文字が有効な値です。

戻り値

このコマンドには、戻り値はありません。

例

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

AnsiClearSetSndRsp

コマンド書式

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

コマンドの説明

このコマンドは、データを送信するときに応答を予期するよう設定します。すべてのシステムが応答を送信するわけではないため、既定では応答を予期しません。

パラメータ

Response	予期する応答が入ります。1 つ以上の文字が有効な値です。

戻り値

このコマンドには、戻り値はありません。

例

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

AnsiClearSetTerm

コマンド書式

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

コマンドの説明

このコマンドは、送受信に使用する終了文字を設定します。既定値は、[ASCII carriage return (0x0D)] です。

パラメータ

Terminator	終了文字が入ります。

戻り値

このコマンドには、戻り値はありません。

例

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

AnsiClearSndAtm

コマンド書式

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

コマンドの説明

このコマンドは、AnsiClear プロトコルを使ってメールボックス添付を送信するために使用します。

パラメータ

MsgId	**MbxGetNextMsg** から返されたメールボックス メッセージ識別子が入ります。
AtmId	**MbxGetNextAtm** から返されたメールボックス添付識別子が入ります。

戻り値

このコマンドには、戻り値はありません。

例

```
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

コマンド書式

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

コマンドの説明

このコマンドは、指定の文字列またはバイト数が検出されるまで ASCII プロトコルを使ってデータを受信します。

パラメータ

StopRcv	いつ受信を中止するのかを示すために使用する制御文字列が入ります。
ByteCount	受信するバイト数が入ります。
Timeout	オプション。既定値は 60 秒です。既定値を上書きするために使用します。
ContinueOnTimeout	オプション。既定値は 0 です。受信中止条件の前にタイムアウトが発生しても処理を続行させる場合は、1 に設定します。

戻り値

このコマンドには、戻り値はありません。

例

```
//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

コマンド書式

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

コマンドの説明

このコマンドは、ASCII プロトコルを使ってデータを受信するために使用します。このコマンドを **SetRcvNoData** および **SetRcvError** と連携して使うと、受信ファイルの代わりに、または受信ファイルがない場合に、ほかのデータ応答を探すことができます。

パラメータ

FileSpec	ファイルを指す fully qualified filespec です。UNC (汎用命名規則) を適用できます。
EndOfFile	ファイルの終わりに達したことを示すために使用します。
Terminate	オプション。既定値は 0 です。1 に設定した場合には、RcvNoData 条件または RcvError 条件が発生したときに受信を終了します。

戻り値

0	ファイルの終わり条件が発生した場合
1	RcvNoData 条件が発生した場合
2	RcvError 条件が発生した場合

例

```
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

コマンド書式

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

コマンドの説明

このコマンドは、ASCII プロトコルを使ってデータを受信するために使用します。受信データは、1 つの添付と一緒にメールボックス メッセージに格納されます。このコマンドを **SetRcvNoData** および **SetRcvError** と連携して使うと、受信ファイルの代わりに、または受信ファイルがない場合に、ほかのデータ応答を探すことができます。

パラメータ

EndOfFile	ファイルの終わりに達したことを示すために使用します。
Terminate	オプション。既定値は 0 です。1 に設定した場合には、**RcvNoData** 条件または **RcvError** 条件が発生したときに受信を終了します。

戻り値

0	ファイルの終わり条件が発生した場合
1	**RcvNoData** 条件が発生した場合
2	**RcvError** 条件が発生した場合

例

```
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

コマンド書式

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

コマンドの説明

このコマンドは、ASCII プロトコルを使ってメールボックス添付を送信するために使用します。

パラメータ

MsgId	**MbxGetNextMsg** から返されたメールボックス メッセージ識別子が入ります。
AtmId	**MbxGetNextAtm** から返されたメールボックス添付識別子が入ります。
EndOfFile	ファイルの終わりを示すために送信されるデータが入ります。

戻り値

このコマンドには、戻り値はありません。

例

```
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

コマンド書式

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

コマンドの説明

このコマンドは、ASCII プロトコルを使って指定の文字列を伝送します。

パラメータ

Data	伝送される制御文字列が入ります。

戻り値

このコマンドには、戻り値はありません。

例

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

AsciiSndFile

コマンド書式

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

コマンドの説明

このコマンドは、ASCII プロトコルを使ってファイルを送信するために使用します。

パラメータ

FileSpec	ファイルを指す fully qualified filespec です。UNC (汎用命名規則) を適用できます。
EndOfFile	ファイルの終わりを示すために送信されるデータが入ります。

戻り値

このコマンドには、戻り値はありません。

例

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

atoi

コマンド書式

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

コマンドの説明

atoi 関数は、文字列を整数に変換する数値関数です。数値関数を使うと、特定のデータ型を別のデータ型に変換できます。

パラメータ

string	文字列変数です。

戻り値

この関数は、文字列の整数値を返します。

例

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

aton

コマンド書式

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

コマンドの説明

aton 関数は、文字列を実数に変換する数値関数です。数値関数を使うと、特定のデータ型を別のデータ型に変換できます。

パラメータ

string	実数に変換される文字列です。

戻り値

この関数は、文字列の実数値を返します。

例

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

Begin...End

コマンド書式

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

コマンドの説明

begin...end キーワードは、**if...then** ループまたは **while...do** ループの本体に複数のステートメントを入れるために使用します。

メモ
ループ内でステートメントを 1 つしか使わない場合は、キーワードの **begin** と **end** を省略できます。

パラメータ

このコマンドには、パラメータはありません。

戻り値

このコマンドには、戻り値はありません。

例

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

BisyncAutoAnswer

コマンド書式

```
BisyncAutoAnswer();
```

コマンドの説明

このコマンドは、バイナリ同期デバイスを受信コールを待機している自動応答モードにします。これは、バイナリ同期デバイスのホスト プール スクリプトでのみ使用します。

パラメータ

このコマンドには、パラメータはありません。

戻り値

このコマンドには、戻り値はありません。

BisyncClose

コマンド書式

```
BisyncClose();
```

コマンドの説明

このコマンドは、現在のトランザクションを終了するためにすべての **BisyncRead** コマンドや **BisyncWrite** コマンドの完了後に発行します。

パラメータ

このコマンドには、パラメータはありません。

戻り値

このコマンドには、戻り値はありません。

例

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

BisyncConfig

コマンド書式

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

コマンドの説明

このコマンドを使うと、現在の構成ファイルを上書きする構成ファイルを読み込むことができます。

パラメータ

ConfigFile	新しい構成ファイル名が入ります。

戻り値

このコマンドには、戻り値はありません。

BisyncDial

コマンド書式

```
BisyncDial();
```

コマンドの説明

このコマンドは、プロパティに格納された電話番号をダイヤルします。このコマンドを使うと、ダイヤルする前に **(BisyncConfig** を使って **)** 新しい構成ファイルを読み込むことができます。システムが **BisyncConfig** を呼び出す前に自動的にダイヤルする場合、ラインが切断されます。

パラメータ

このコマンドには、パラメータはありません。

戻り値

このコマンドには、戻り値はありません。

例

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

BisyncEot

コマンド書式

```
BisyncEot();
```

コマンドの説明

このコマンドを使うと、伝送終了文字を送ってラインを無条件でクリアすることができます。通常は **BisyncClose** コマンドの後に続けます。

パラメータ

このコマンドには、パラメータはありません。

戻り値

このコマンドには、戻り値はありません。

例

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

BisyncOpen

コマンド書式

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

コマンドの説明

このコマンドは、指定された定数に基づいてデータを送信または受信するためにラインを開きます。受信用のラインを開く際には、ライン ビッドを待って受信確認で応答します。送信用のラインを開く際には、ラインをビッドして受信確認を待ちます。

パラメータ

Mode　次のオープン モード定数の 1 つが入ります。

定数	説明
READ_TEXT	非透過テキスト データをトランスレーションによって受信できるようにします。透過バイナリ データは、トランスレーションなしで受信されます。
READ_NO_TRANSLATE	非透過テキスト データをトランスレーションなしで受信できるようにします。
READ_BINARY_TRANSLATE	透過バイナリ データをトランスレーションによって受信できるようにします。
WRITE_TEXT	非透過テキスト データをトランスレーションによって送信できるようにします。
WRITE_BINARY	透過バイナリ データをトランスレーションなしで送信できるようにします。
WRITE_TEXT_NO_TRANSLATE	非透過テキスト データをトランスレーションなしで送信できるようにします。
WRITE_NO_IRS	非透過テキスト データをトランスレーションによって送信できるようにしますが、レコード セパレータは含めません。これは実質的に、データを 1 つのレコードにパッキングします。
WRITE_BINARY_TRANSLATE	透過バイナリ データをトランスレーションによって送信できるようにします。

(次のページへ続く)

(続き) 定数	説明
WRITE_TEXT_BINARY	透過バイナリ データをトランスレーションによって送信できるようにしますが、特殊文字の特殊トランスレーション (CR/LF から RS) を伴います。

戻り値

このコマンドには、戻り値はありません。

例

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

BisyncRcvCtl

コマンド書式

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

コマンドの説明

このコマンドは、受信データの内容チェックに使用可能な内部受信バッファにデータを受け取るために使用します。いつ受信を中止するのかを示す別のプロトコル文字を指定し、受信を終了したプロトコル文字の値を返します。

パラメータ

EndRcv	いつ受信を中止するのかを示すプロトコル文字を指定します。
1	EOT を受け取ったときにデータの受信を中止します。
2	ETB を受け取ったときにデータの受信を中止します。
3	ETX を受け取ったときにデータの受信を中止します。
4	ETB または ETX を受け取ったときにデータの受信を中止します。 戻り値コードは、受信を中止させた内容によって決まります。

戻り値

1	EOT が受信されました。
2	ETB が受信されました。
3	ETX が受信されました。

(次のページへ続く)

例

```
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");
```

関連トピック

このコマンドの追加書式については、BisyncRcvCtl (integer EndRcv) スクリプトを参照してください。

BisyncRcvFile

コマンド書式

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

コマンドの説明

このコマンドは、Bisync プロトコルを使ってファイルを受信ために使用します。このコマンドを **SetRcvNoData** および **SetRcvError** と連携して使うと、受信ファイルの代わりに、または受信ファイルがない場合に、ほかのデータ応答を探すことができます。

パラメータ

FileSpec	ファイルを指す fully qualified filespec です。UNC (汎用命名規則) を適用できます。

戻り値

0	通常のファイルの終わり
1	RcvNoData 条件が発生した場合
2	RcvError 条件が発生した場合

例

```
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

コマンド書式

```
integer BisyncRcvMsg();
```

コマンドの説明

このコマンドは、Bisync プロトコルを使ってデータを受信ために使用します。受信データは、1 つの添付と一緒にメールボックス メッセージに入れられます。このコマンドを **SetRcvNoData** および **SetRcvError** と連携して使うと、受信ファイルの代わりに、または受信ファイルがない場合に、ほかのデータ応答を探すことができます。

パラメータ

このコマンドには、パラメータはありません。

戻り値

0	通常のファイルの終わり
1	**RcvNoData** 条件が発生した場合
2	**RcvError** 条件が発生した場合

例

```
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

コマンド書式

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

コマンドの説明

このコマンドは、送受信の両方で使う送信終了文字を設定します。

パラメータ

EofChar	ファイル終了文字が入ります。

戻り値

このコマンドには、戻り値はありません。

例

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

BisyncSndAtm

コマンド書式

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

コマンドの説明

このコマンドは、Bisync プロトコルを使ってメールボックス添付を送信するために使用します。

パラメータ

MsgId	**MbxGetNextMsg** から返されたメールボックス メッセージ識別子が入ります。
AtmId	**MbxGetNextAtm** から返されたメールボックス添付識別子が入ります。

戻り値

このコマンドには、戻り値はありません。

例

```
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

コマンド書式

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

コマンドの説明

このコマンドでは、Bisync プロトコルを使って指定された文字列を伝送します。

パラメータ

Data	伝送される制御文字列が入ります。

戻り値

このコマンドには、戻り値はありません。

例

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

BisyncSndCtlEtx

コマンド書式

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

コマンドの説明

このコマンドでは、Bisync プロトコルを使って指定の文字列を伝送し、各ブロックを ETX (end of text) で終了します。

パラメータ

Data	伝送される制御文字列が入ります。

戻り値

このコマンドには、戻り値はありません。

例

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

BisyncSndFile

コマンド書式

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

コマンドの説明

このコマンドは、Bisync プロトコルを使ってファイルを送信するために使用します。

パラメータ

FileSpec	ファイルを指す fully qualified filespec です。UNC (汎用命名規則) を適用できます。

戻り値

このコマンドには、戻り値はありません。

例

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

BisyncTable

コマンド書式

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

コマンドの説明

このコマンドでは、新しいトランスレーション テーブルをロードできます。新しいテーブルは、asciiebc.ovr ファイルや ebcascii.ovr ファイルと同じ形式でなければなりません。

パラメータ

Table	スペース文字で区切ったファイル名が入ります。

戻り値

このコマンドには、戻り値はありません。

例

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

Break

コマンド書式

```
break;
```

コマンドの説明

break キーワードは、最も近くでそれを囲む **while** ループの実行を終了し、**end** キーワードの後ろのステートメントに制御を渡します。**break** キーワードは、通常では複雑なループ内で使用し、いくつかのステートメントの実行前にそのループを終了します。

パラメータ

このコマンドには、パラメータはありません。

戻り値

このコマンドには、戻り値はありません。

例

```
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

コマンド書式

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

コマンドの説明

concat 関数は、1 つの文字列の指定数の文字を別の文字列の末尾に連結します。

パラメータ

string1	最初の文字列の変数名です。
string2	2 番目の文字列の変数名です。
num_char	最初の文字列の末尾に連結する 2 番目の文字列の文字数です。

戻り値

このコマンドには、戻り値はありません。

例

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

Continue

コマンド書式

```
continue;
```

コマンドの説明

continue キーワードは、**continue** ステートメントの後ろのループ内のステートメントを処理せずに、最も内側のループの処理を続行します。

パラメータ

このコマンドには、パラメータはありません。

戻り値

このコマンドには、戻り値はありません。

例

```
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

コマンド書式	`DclLogoff();`
コマンドの説明	このコマンドは、DCL プロトコルを使ってログオフ手順を実行します。
パラメータ	このコマンドには、パラメータはありません。
戻り値	このコマンドには、戻り値はありません。

DclLogon

コマンド書式

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

コマンドの説明

このコマンドは、DCL プロトコルを使ってログオン手順を実行します。

パラメータ

LogonAcct	Advantis ログオン アカウントが入ります。
LogonId	Advantis ログオン識別子が入ります。
LogonPsw	Advantis ログオン パスワードが入ります。
IEAcct	情報交換アカウントが入ります。
IEId	情報交換識別子が入ります。
IEPsw	情報交換パスワードが入ります。
NewLogonPsw	新しい Advantis ログオン パスワードが入ります。これを使うのは、ログオン パスワードを変更する場合のみです。
NewIEPsw	新しい情報交換パスワードが入ります。これを使うのは、情報交換パスワードを変更する場合のみです。

戻り値

このコマンドには、戻り値はありません。

DclRcvMsg

コマンド書式

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

コマンドの説明

このコマンドは、DCL プロトコルを使ってデータを受信するために使用します。受信データは、1 つの添付と一緒にメールボックス メッセージに入れられます。

パラメータ

UserMsgClass	オプション。これを指定すると、指定されたユーザー メッセージ分類を保持するデータを取得します。

戻り値

このコマンドには、戻り値はありません。

DclSetAck

コマンド書式

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

コマンドの説明

このコマンドは、Information Exchange から受け取る受信確認メッセージの種類を設定するために使用します。

パラメータ

AckType	受け取る受信確認メッセージの種類です。有効な値は次のとおりです。 [R] — 受信確認 [D] — 配送確認 [B] — 受信と配送の両方

戻り値

このコマンドには、戻り値はありません。

例

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

DclSndAtm

コマンド書式

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

コマンドの説明

このコマンドは、DCL プロトコルを使ってメールボックス添付を送信するために使用します。

パラメータ

MsgId	**MbxGetNextMsg** から返されたメールボックス メッセージ識別子が入ります。
AtmId	**MbxGetNextAtm** から返されたメールボックス添付識別子が入ります。

戻り値

このコマンドには、戻り値はありません。

例

```
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

コマンド書式

```
integer DoRcv
```

コマンドの説明

このコマンドは、受信型セッション (受信専用または送受信) が要求されたかどうかを判断するために使用することができます。スクリプトはメールボックス定義ごとに 1 つしかないため、このコマンドは要求されたセッションの種類を示して、適切なコマンドを発行できるようにします。

パラメータ

このコマンドには、パラメータはありません。

戻り値

0	受信専用セッションや送受信型セッションでない場合
1	受信専用セッションや送受信型セッションの場合

例

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

DoSnd

コマンド書式

```
integer DoSnd
```

コマンドの説明

このコマンドは、送信型セッション (送信専用または送受信) が要求されたかどうかを判断するために使用することができます。スクリプトはメールボックス定義ごとに 1 つしかないため、このコマンドは要求されたセッションの種類を示して、適切なコマンドを発行できるようにします。

パラメータ

このコマンドには、パラメータはありません。

戻り値

0	送信専用セッションや送受信型セッションでない場合
1	送信専用セッションや送受信型セッションの場合

例

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

EiconCall

コマンド書式

```
EiconCall();
```

コマンドの説明

このコマンドは、Eicon Technology ハードウェアを使って X.25 接続を行うために使用します。

パラメータ

このコマンドには、パラメータはありません。

戻り値

このコマンドには、戻り値はありません。

EiconListen

コマンド書式

```
EiconListen();
```

コマンドの説明

このコマンドは、Eicon Technology ハードウェアを使ってアドバンスド データ ディストリビューションで X.25 接続を受信するために使用します。

パラメータ

このコマンドには、パラメータはありません。

戻り値

このコマンドには、戻り値はありません。

EiconSetBICUG

コマンド書式

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

コマンドの説明

このコマンドは、二者間で閉じたユーザー グループ選択に X.25 機能を設定するために使用します。詳細な説明と値の設定については、ITU (国際電気通信連合) の X.25 勧告 (1984 年) を参照してください。

パラメータ

Value1	二者間の閉じたユーザー グループを示す 1 番目と 2 番目の数字のインデックス番号です。
Value2	二者間の閉じたユーザー グループを示す 3 番目と 4 番目の数字のインデックス番号です。

戻り値

このコマンドには、戻り値はありません。

EiconSetClassNeg

コマンド書式

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

コマンドの説明

このコマンドは、X.25 機能のスループット クラス ネゴシエーションを設定するために使用します。詳細な説明と値の設定については、ITU (国際電気通信連合) の X.25 勧告 (1984 年) を参照してください。

パラメータ

Value	スループット クラス値

戻り値

このコマンドには、戻り値はありません。

例

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

EiconSetCUG

コマンド書式

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

コマンドの説明

このコマンドは、X.25 機能の閉じたユーザー グループ選択を設定するために使用します。詳細な説明と値の設定については、ITU (国際電気通信連合) の X.25 勧告 (1984 年) を参照してください。

パラメータ

Value	閉じたユーザー グループのインデックス番号

戻り値

このコマンドには、戻り値はありません。

EiconSetCUGOA

コマンド書式

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

コマンドの説明

このコマンドは、X.25 機能の閉じたユーザー グループをアウトバウンド アクセス選択で設定するために使用されます。詳細な説明と値の設定については、ITU (国際電気通信連合) の X.25 勧告 (1984 年) を参照してください。

パラメータ

Value	送信側の閉じたユーザー グループのインデックス番号

戻り値

このコマンドには、戻り値はありません。

EiconSetNUI

コマンド書式

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

コマンドの説明

このコマンドは、ネットワーク ユーザー識別の X.25 機能を設定するために使用します。詳細な説明と値の設定については、ITU (国際電気通信連合) の X.25 勧告 (1984 年) を参照してください。

パラメータ

Length	NUI 文字列の長さです。
NUI	パスワード最大長の 6 バイトと最大 8 バイト長のデータから構成される NUI の内容です。

戻り値

このコマンドには、戻り値はありません。

EiconSetPacketSize

コマンド書式

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

コマンドの説明

このコマンドは、X.25 機能のパケット サイズを設定するために使用します。詳細な説明と値の設定については、ITU (国際電気通信連合) の X.25 勧告 (1984 年) を参照してください。

パラメータ

Remote	リモートのパケット サイズ コード
Local	ローカルのパケット サイズ コード

戻り値

このコマンドには、戻り値はありません。

例

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

EiconSetRevFast

コマンド書式

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

コマンドの説明

このコマンドは、着信課金または高速選択、あるいは両方のために X.25 機能を設定するために使用されます。詳細な説明と値の設定については、ITU (国際電気通信連合) の X.25 勧告 (1984 年) を参照してください。

パラメータ

Value	着信課金または高速選択、あるいは両方の値

戻り値

このコマンドには、戻り値はありません。

EiconSetUserData

コマンド書式

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

コマンドの説明

このコマンドは、X.29 のコール ユーザー データ パラメータを設定するために使用します。

パラメータ

P1	プロトコル ID 1
P2	プロトコル ID2
P3	プロトコル ID 3
P4	プロトコル ID 4
UserData	ユーザー定義のデータ (12 バイト以内)

戻り値

このコマンドには、戻り値はありません。

例

```
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

コマンド書式

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

コマンドの説明

このコマンドは、X.25 機能のウィンドウ サイズを設定するために使用します。詳細な説明と値の設定については、ITU (国際電気通信連合) の X.25 勧告 (1984 年) を参照してください。

パラメータ

Remote	リモートのウィンドウ サイズ
Local	ローカルのウィンドウ サイズ

戻り値

このコマンドには、戻り値はありません。

例

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

FtpCD

コマンド書式

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

コマンドの説明

このコマンドは、通常の CWD コマンドの代わりに CD コマンドを使って FTP サーバー上のディレクトリを変更するために使用します。GEIS によって使用されます。

パラメータ

Directory	FTP サーバー上で変更するディレクトリ

戻り値

このコマンドには、戻り値はありません。

FtpChangeDir

コマンド書式

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

コマンドの説明

このコマンドは、FTP サーバー上のディレクトリを変更するために使用します。FtpChangeDir は、まず最初に CWD コマンドを発行し、そのコマンドが失敗して 502 エラーを受信した場合に CD コマンドを使用して再試行します。

パラメータ

Directory	FTP サーバー上で変更するディレクトリ

戻り値

このコマンドには、戻り値はありません。

FtpDelete

コマンド書式

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

コマンドの説明

このコマンドは、FTP サーバー上のファイルを削除するために使用します。

パラメータ

FileName	FTP サーバー上で削除するファイルの名前

戻り値

このコマンドには、戻り値はありません。

FtpDoCmd

コマンド書式

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

コマンドの説明

このコマンドは、RFC 準拠の FTP コマンドを発行するために使用します。

パラメータ

Cmd	発行する RFC 準拠の FTP コマンド
Arg	FTP コマンドの引数。オプション。

戻り値

ホストから返された最終行の先頭で検出された値を返します。ただし、500 より大きい値の場合にはエラーが発生します。ユーザー自身がこのコードを評価し、コマンドが成功したかどうかを判断します。

例

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

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

FtpGetDir

コマンド書式

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

コマンドの説明

このコマンドは、FTP サーバーからディレクトリ リストを取得するために使用します。このコマンドを FtpGetDirString と連携して使うと、サーバーから返されるディレクトリ出力の各行を検査できます。

パラメータ

bUseList	ディレクトリ リストの種類を示すブール変数です。値が [TRUE] の場合、LIST コマンドがサーバーに送られます。値が [FALSE] の場合、NLST コマンドがサーバーに送られます。

戻り値

返される整数値は、サーバーから返されたディレクトリの列数 / 行数を示します。

例

```
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

コマンド書式

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

コマンドの説明

このコマンドは、FTP サーバーから返されるディレクトリ出力の各行を返すために使用します。ディレクトリ出力を受け取るには、この関数を使う前に FtpGetDir への呼出しが発生しなければなりません。

パラメータ

iIndex	ディレクトリ 出力配列のゼロ ベースのインデックスです。
buffer	配列インデックスに基づいて、そのディレクトリ行の内容を入れる文字列バッファです。

戻り値

このコマンドには、戻り値はありません。

例

```
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

コマンド書式

```
FtpHost();
```

コマンドの説明

このコマンドは、FTP サーバー セッションを開始するために使用します。

パラメータ

このコマンドには、パラメータはありません。

戻り値

このコマンドには、戻り値はありません。

例

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

FtpRcvFile

コマンド書式

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

コマンドの説明

このコマンドは、FTP プロトコルを使ってファイルを受信するために使用します。このコマンドを **SetRcvNoData** および **SetRcvError** と連携して使うと、ファイルの代わりに、またはファイルがない場合に、ほかの応答を探すことができます。

パラメータ

LocalFile	ローカル システムで作成するファイルの名前です。
RemoteFile	リモート システム上のファイルの名前です。
DeleteAfterRcv	オプション。既定値は、ファイルを削除しない 0 または [FALSE] です。1 または [TRUE] に設定すると、受信の完了後に各ファイルを削除できます。

戻り値

0	受信が正常に完了した場合
1	RcvNoData 条件が発生した場合
2	RcvError 条件が発生した場合

例

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

FtpRcvMsg

コマンド書式

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

コマンドの説明

このコマンドは、FTP プロトコルを使ってファイルを 1 つ受信するために使用します。受信データは、1 つの添付と一緒にメールボックス メッセージに入れられます。

パラメータ

FileName	FTP サーバーから受信するファイルの名前
DeleteAfterRcv	オプション。既定値は、ファイルを削除しない 0 または [FALSE] です。1 または [TRUE] に設定すると、受信の完了後に各ファイルを削除できます。

戻り値

このコマンドには、戻り値はありません。

例

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

FtpRcvMsgAll

コマンド書式

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

コマンドの説明

このコマンドは、FTP サーバー上の現在のディレクトリからすべてのファイルを受信するために使用します。受信データは、1 つの添付と一緒にメールボックスメッセージに入れられます。

パラメータ

DeleteAfterRcv	オプション。既定値は、ファイルを削除しない 0 または FALSE です。1 または [TRUE] に設定すると、受信の完了後に各ファイルを削除できます。

戻り値

このコマンドには、戻り値はありません。

例

```
FtpRcvMsgAll(TRUE);
```

FtpRename

コマンド書式

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

コマンドの説明

このコマンドは、FTP サーバー上のファイルの名前を変更するために使用します。

パラメータ

OldName	名前を変更する既存のファイルの名前を指定します。
NewName	新しいファイル名を指定します。

戻り値

このコマンドには、戻り値はありません。

例

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

FtpSetMode

コマンド書式

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

コマンドの説明

このコマンドは、FTP サーバーにファイルを転送するモードを設定するために使用します。

パラメータ

Mode	[ASCII] モードまたは [BINARY] モードを指定します。既定値は [BINARY] モードです。

戻り値

このコマンドには、戻り値はありません。

例

```
FtpSetMode (ASCII);
```

FtpSndAtm

コマンド書式

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

コマンドの説明

このコマンドは、FTP プロトコルを使ってメールボックス添付を送信するために使用します。

パラメータ

MsgId	**MbxGetNextMsg** から返されたメールボックス メッセージ識別子が入ります。
AtmId	**MbxGetNextAtm** から返されたメールボックス添付識別子が入ります。
FileName	FTP サーバー上で作成するファイルの名前が入ります。

戻り値

このコマンドには、戻り値はありません。

例

```
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

コマンド書式

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

コマンドの説明

このコマンドは、FTP トランスポートまたは WSFTP トランスポートを使ってメールボックス システム内にないファイルを FTP サーバーに送信するために使用します。

パラメータ

LocalFilename	この変数は、送信するファイルのローカル ファイル名を指定します。
RemoteFilename	この変数は、ファイルが送信されたときに FTP サーバー上に作成されるファイルのリモート ファイル名を指定します。
DeleteFlag	ローカル ファイルが正常に送信された後でそのファイルを削除するかどうかを指定します。有効な値は次のとおりです。[TRUE] (ローカル ファイルの正常な送信後にそのファイルを削除) または [FALSE] (ローカル ファイルの正常な送信後にそのファイルを削除しない)。この変数は、TRUE または FALSE で指定しなければなりません。既定値はありません。

戻り値

このコマンドには、戻り値はありません。

get

コマンド書式

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

コマンドの説明

get 関数では、日付 / 時刻変数の個々のコンポーネントを取得できます。

パラメータ

datetime_component	日付 / 時刻変数の個々のコンポーネントです。
datetime_variable	アクセスするコンポーネントの日付 / 時刻変数です。

戻り値

この関数は、datetime_component の整数値を返します。

例

```
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

コマンド書式

```
integer GetSessionTry();
```

コマンドの説明

このコマンドは、現在実行中のセッション試行が初期試行であるか、または再試行であるかを判別するために使用します。

戻り値

初期セッション試行なら、戻り値は 0 です。各セッションの再試行の場合、戻り値は 0 よりも大きくなります。

例

```
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

コマンド書式

```
if condition then
```

コマンドの説明

キーワードの **if**、**then**、および **else** を使うと、スクリプトで条件付き論理を利用できます。Gentran:Server では、条件付き論理を使って条件をテストし、そのテスト結果に応じて演算を実行します。条件は、任意のレベルに対してネストすることができます。通常、条件には比較を使用しますが、数値で終了する任意の式を使用することもできます。Gentran:Server は、*true* と *false* のいずれかとして値を解釈します。システムでは、ゼロ値が [false]、ゼロ以外の値が [true] と解釈されます。

Gentran:Server では、**if...then** 条件を評価し、それが true であれば **then** キーワードに続くすべてのステートメントが実行されます。その条件が false であれば、**then** に続くステートメントは実行されません。

if...then と連携して **else** キーワードを使うと、複数ブロックにわたるステートメントを定義することができ、結果的にそれらのブロックのうちの 1 つが実行されます。Gentran:Server では、まず **if...then** 条件がテストされます。その条件が false であれば、true である条件が検出されるまで各条件のテストが順次進められます。true の条件があれば、それに対応するブロックのステートメントが実行されます。**if...then** 条件に true のものがなければ、**else** キーワードに続くステートメントが実行されます。

パラメータ

このコマンドには、パラメータはありません。

戻り値

このコマンドには、戻り値はありません。

例

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

KermitRcvMsg

コマンド書式

```
KermitRcvMsg();
```

コマンドの説明

このコマンドは、Kermit プロトコルを使ってデータを受信するために使用します。

パラメータ

このコマンドには、パラメータはありません。

戻り値

このコマンドには、戻り値はありません。

例

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

KermitSndAll

コマンド書式

```
KermitSndAll();
```

コマンドの説明

このコマンドは、Kermit プロトコルを使ってすべてのファイルを送信するために使用します。

パラメータ

このコマンドには、パラメータはありません。

戻り値

このコマンドには、戻り値はありません。

例

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

KermitSet8thBitQuote

コマンド書式

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

コマンドの説明

文字 (10 進表記) を 8 番目のビット引用文字に使うよう指定します。このコマンドは、既定の 8thBitQuote 値を上書きするために使用します。

パラメータ

8thBitQuote	8 番目のビット引用文字として使う 10 進表記の文字。これは、33 ～ 126 の範囲の値です。既定値は 0 で、8 番目のビット引用文字を使いません。

戻り値

このコマンドには、戻り値はありません。

KermitSetBlockCheckType

コマンド書式

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

コマンドの説明

使用するブロック チェックの種類を指定します。このコマンドは、既定の BlockCheckType 値を上書きするために使用します。

パラメータ

BlockCheckType	1 = 1 バイトのチェックサム 2 = 2 バイトのチェックサム 3 = 3 バイトの CRC 既定値は 1 です。

戻り値

このコマンドには、戻り値はありません。

KermitSetBlockStart

コマンド書式

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

コマンドの説明

文字 (10 進表記) をブロック開始を示すために使うよう指定します。このコマンドは、既定の BlockStart 値を上書きするために使用します。

パラメータ

BlockStart	ブロック開始を示すために使う 10 進表記の文字。これは、0 〜 127 の範囲の値です。既定値は 1 (SOH) です。

戻り値

このコマンドには、戻り値はありません。

KermitSetControlQuote

コマンド書式

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

コマンドの説明

文字 (10 進表記) を引用制御を示すために使うよう指定します。このコマンドは、既定の ControlQuote 値を上書きするために使用します。

パラメータ

ControlQuote	引用制御を示すために使う 10 進表記の文字。これは、32 ～ 127 の範囲の値です。既定値は 35 (#) です。

戻り値

このコマンドには、戻り値はありません。

KermitSetEndOfLine

コマンド書式

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

コマンドの説明

行の終わりに使用する文字 (10 進表記) を指定します。このコマンドは、既定の EndOfLine 値を上書きするために使用します。

パラメータ

EndOfLine	行の終わりに使用する 10 進表記の文字。これは、0 ～ 127 の範囲の値です。既定値は 13 (CR) です。

戻り値

このコマンドには、戻り値はありません。

KermitSetNumberOfPads

コマンド書式

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

コマンドの説明

使用する付け足し文字の数を指定します。このコマンドは、既定の NumberOfPads 値を上書きするために使用します。

パラメータ

NumberOfPads	使用する付け足し文字の数。これは、0 ～ 127 の範囲の値です。既定値は、文字数ゼロを示す 0 になります。

戻り値

このコマンドには、戻り値はありません。

KermitSetPacketSize

コマンド書式

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

コマンドの説明

ファイル転送で使う最大のパケット サイズを指定します。このコマンドは、既定の PacketSize 値を上書きするために使用します。

パラメータ

PacketSize	ファイル転送時における Kermit データ パケットの最大サイズが入ります。既定値は 1024 です。

戻り値

このコマンドには、戻り値はありません。

KermitSetPadCharacter

コマンド書式

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

コマンドの説明

文字 (10 進表記) を付け足し文字として使うよう指定します。このコマンドは、既定の PadCharacter 値を上書きするために使用します。

パラメータ

PadCharacter	付け足し文字として使う 10 進表記の文字。これは、0 ～ 127 の範囲の値です。既定値は 0 (NULL) です。

戻り値

このコマンドには、戻り値はありません。

KermitSetParity

コマンド書式

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

コマンドの説明

そのパリティは使用中であるため考慮に入れる必要があることを示します。このコマンドは、既定の Parity 値を上書きするために使用します。

パラメータ

Parity	0 (ゼロ) または 1 です。1 に設定すると、パリティを使うために特殊な処理が必要となることを Kermit プロトコルに通知できます。既定値は、パリティなしを示す 0 です。

戻り値

このコマンドには、戻り値はありません。

left

コマンド書式

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

コマンドの説明

left 関数は、文字列変数または文字列フィールド内の左側から指定した文字数を抽出し、その結果を文字列として返します。

パラメータ

string_variable	型文字列として定義された変数です。
num_char	文字列の左側からカウントする文字数です。

戻り値

この関数は、変数の文字列値を返します。

例

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

len

コマンド書式

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

コマンドの説明

len 関数は、文字列中の文字数を数えて戻す数値関数です。数値関数を使うと、特定のデータ型を別のデータ型に変換できます。

パラメータ

string	評価する文字列

戻り値

この関数は、文字列の長さを返します。

例

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

LogMessage

コマンド書式

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

コマンドの説明

このコマンドは、ユーザー指定の文字列またはメッセージをセッション ログに書き込むために使用します。

パラメータ

UserMsg	セッション ログに書き込む文字列値

戻り値

このコマンドには、戻り値はありません。

例

```
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

コマンド書式

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

コマンドの説明

このコマンドは、添付からコンテンツ タイプとコンテンツ サブタイプを取得するために使用します。

パラメータ

AtmId	照会する添付の識別子が入ります。
ContentType	コンテンツ タイプを返すために使う文字列値
ContentSubType	コンテンツ サブタイプを返すために使う文字列値

戻り値

このコマンドには、戻り値はありません。

例

```
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

コマンド書式

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

コマンドの説明

このコマンドは、メッセージ添付のファイル拡張子を取得するために使用します。添付のファイル パスが C:\TEST\TEXT.TXT の場合、この関数は TXT を返します。

パラメータ

MsgId	**MbxGetNextMsg** から返されたメールボックス メッセージ識別子が入ります。
AtmId	**MbxGetNextAtm** から返されたメールボックス添付識別子が入ります。
FileExt	添付のファイル拡張子を返すために使われる文字列値です。

戻り値

このコマンドには、戻り値はありません。

例

```
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

コマンド書式

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

コマンドの説明

このコマンドは、メッセージ添付のファイル名を取得するために使用します。添付のファイル パスが C:¥TEST¥TEXT.TXT の場合、この関数は TEXT.TXT を返します。

パラメータ

MsgId	**MbxGetNextMsg** から返されたメールボックス メッセージ識別子が入ります。
AtmId	**MbxGetNextAtm** から返されたメールボックス添付識別子が入ります。
Filename	添付のファイル名を返すために使われる文字列値です。

戻り値

このコマンドには、戻り値はありません。

例

```
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

コマンド書式

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

コマンドの説明

このコマンドは、メッセージ添付のファイル パスを取得するために使用します。添付のファイル パスが C:¥TEST¥TEXT.TXT の場合、この関数は C:¥TEST¥TEXT.TXT を返します。

パラメータ

MsgId	**MbxGetNextMsg** から返されたメールボックス メッセージ識別子が入ります。
AtmId	**MbxGetNextAtm** から返されたメールボックス添付識別子が入ります。
FilePath	添付のファイル パスを返すために使われる 文字列値です。

戻り値

このコマンドには、戻り値はありません。

例

```
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

コマンド書式

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

コマンドの説明

このコマンドは、メッセージ添付のファイル タイトルを取得するために使用します。添付のファイル パスが C:¥TEST¥TEXT.TXT の場合、この関数は TEXT を返します。

パラメータ

MsgId	**MbxGetNextMsg** から返されたメールボックス メッセージ識別子が入ります。
AtmId	**MbxGetNextAtm** から返されたメールボックス添付識別子が入ります。
FileTitle	添付のファイル タイトルを返すために使われる文字列値です。

戻り値

このコマンドには、戻り値はありません。

例

```
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

コマンド書式

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

コマンドの説明

このコマンドは、送信準備が整っている次のメールボックス添付識別子を返します。

パラメータ

AtmId	この変数は戻り値として使用します。

戻り値

ゼロ以外	有効なメールボックス添付識別子が使用可能で、送信準備が整っている場合
ゼロ	送信できるメールボックス添付がそれ以上ない場合

例

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

MbxGetNextMsg

コマンド書式

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

コマンドの説明

このコマンドは、送信準備が整っている次のメールボックス メッセージ識別子を返します。

パラメータ

MsgId	この変数は戻り値として使用します。

戻り値

ゼロ以外	有効なメールボックス メッセージ識別子が使用可能で、送信準備が整っている場合
ゼロ	送信できるメールボックス メッセージがそれ以上ない場合

例

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

MbxGetOriginalMsgId

コマンド書式

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

コマンドの説明

このコマンドは、メッセージから元のメッセージ識別子を取得するために使用します。

パラメータ

MsgId	照会するメッセージのメールボックス メッセージ識別子が入ります。
OriginalMsgId	その情報を返すために使う文字列値です。

戻り値

このコマンドには、戻り値はありません。

例

```
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

コマンド書式

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

コマンドの説明

このコマンドは、メッセージから受信者のゲートウェイ E メール アドレスを取得するために使用します。

パラメータ

MsgId	照会するメッセージのメールボックス メッセージ識別子が入ります。
RcvrEmailAddr	受信者のゲートウェイ E メール アドレスを返すために使う文字列値です。

戻り値

このコマンドには、戻り値はありません。

(次のページへ続く)

例

```
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

コマンド書式

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

メッセージの説明

このコマンドは、メールボックス ID とアドバンスド データ ディストリビューション パスワードを収集するためにアドバンスド データ ディストリビューションで使用します。

パラメータ

mailbox	アドバンスド データ ディストリビューション メールボックスの名前
password	ホスト パスワード

戻り値

このコマンドには、戻り値はありません。

例

```
// 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

コマンド書式

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

コマンドの説明

このコマンドは、特定メッセージのために送信準備が整っているすべてのメールボックス添付を反復処理するために、**MbxStartMsgLoop**、**MbxGetNextMsg**、および **MbxGetNextAtm** と連携して使用します。このコマンドは、メッセージループの設定後のみ使用することができ、**MbxGetNextAtm** が必ず続きます。

パラメータ

MsgId	**MbxGetNextMsg** から返されたメールボックス メッセージ識別子が入ります。

戻り値

このコマンドには、戻り値はありません。

例

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

MbxStartMsgLoop

コマンド書式

```
MbxStartMsgLoop();
```

コマンドの説明

このコマンドは、送信準備が整っているすべてのメールボックス メッセージを反復処理するために、**MbxGetNextMsg** と連携して使用します。このコマンドは、**MbxGetNextMsg** の処理を続行します。

パラメータ

このコマンドには、パラメータはありません。

戻り値

このコマンドには、戻り値はありません。

例

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

mid

コマンド書式

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

コマンドの説明

mid 関数は、文字列内の指定した位置から、文字列の末尾まで、または指定した文字数を抽出し、その結果得られた文字列を返します。

パラメータ

string_variable	抽出する文字列が入った変数です。
start_pos	文字列内で抽出を開始する位置です。
num_char	抽出を開始する位置からの文字数です。

戻り値

この関数は、パラメータで指定した部分文字列の値を返します。

例

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

ntoa

コマンド書式

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

コマンドの説明

ntoa 関数は、実数を文字列に変換する数値関数です。数値関数を使うと、特定のデータ型を別のデータ型に変換できます。

パラメータ

real	変換する数値または変数です。
string	変換済みの数値を格納する文字列の名前です。

戻り値

このコマンドには、戻り値はありません。

例

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

OftpAddSfidCheck

コマンド書式

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

コマンドの説明

このコマンドは、指定した文字列を、ファイルの受信時に SFID の sfidorig フィールドと照合される有効なオリジネータ ID のテーブルに追加します。ファイルは、sfidorig フィールドの内容がテーブル内のどの値にも一致しない場合には拒否されます。このコマンドは、テーブル内になければならない各 ID に必ず発行され、スクリプト コマンドの **OftpRemote** または **OftpHost** の処理を進行します。

パラメータ

CheckOriginator	SFID sfidorig フィールドに対して照合されるオリジネータ ID です。

戻り値

このコマンドには、戻り値はありません。

例

次に、コマンドの使用例を示します。

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

関連トピック

OFTP プロトコル使用の詳細については、『コミュニケーションズ ゲートウェイ設定 ガイド』または『アドバンスド データ ディストリビューション ゲートウェイ設定 ガイド』の付録の「OFTP の操作」を参照してください。

OftpHost

コマンド書式

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

コマンドの説明

このコマンドは、トレーディング パートナーが OFTP サーバーに対するコミュニケーション セッションを開始したときにアドバンスド データ ディストリビューション機能を実行するために使用します。このコマンドは、メールボックス形式のコマンドを何も指定せずに、要求されたセッションの種類によって、すべての送信や受信を処理します。

パラメータ

SSIDcode	SSID の ssidcode フィールドで送信する文字列値です。
SSIDpwsd	SSID の ssidpwsd フィールドで送信する文字列値です。
SSIDuser	SSID の ssiduser フィールドで送信する文字列値です。
LogonUsingSSID	オプションの値。TRUE に設定すると、MbxLogon スクリプト コマンドではなく受信されたリモート SSID 情報を使ってメールボックスにログオンします。

戻り値

このコマンドには、戻り値はありません。

例 1

次に、OftpHost コマンドの使用例を示します。

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

例 2

ここでは、オプションの LogonUsingSSID 値を指定した OftpHost コマンドの例を示します。

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

OftpNoEerps

コマンド書式

```
OftpNoEerps();
```

コマンドの説明

このコマンドでは、システムに対して、受信されたファイルの EERP を送らないこと、および送信したファイルの EERP を予定しないことを指示します。

パラメータ

このコマンドには、パラメータはありません。

戻り値

このコマンドには、戻り値はありません。

関連トピック

OFTP プロトコル使用の詳細については、『コミュニケーションズ ゲートウェイ設定 ガイド』または『アドバンスド データ ディストリビューション ゲートウェイ設定 ガイド』の付録の「OFTP の操作」を参照してください。

OftpRemote

コマンド書式

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

コマンドの説明

このコマンドは、OFTP サーバーに対して完全なリモート セッションを実行するために使用します。このコマンドは、メールボックス形式のコマンドを何も指定せずに、要求されたセッションの種類によって、すべての送信や受信を処理します。オプションのパラメータを使うと、SSID に収められた ID とパスワードの情報が正確かどうかを確認できます。情報に不一致点が見つかると、セッションは失敗します。オプションのパラメータは、お互いに連携させて使うことが必要です。

パラメータ

OftpId	OFTP ユーザー ID が入ります。
OftpPsw	OFTP ユーザー パスワードが入ります。
OftpNewPsw	OFTP の新しいユーザー パスワードが入ります。これは、パスワードの変更が可能な特定システムでのみ使用します。これを使った場合、ssid 構造の ssiduser フィールドに入ります。
IdToCheck	SSID に収められた OftpId が正確かどうかを確認します。PswToCheck と連携して使うことが必要です。
PswToCheck	SSID に収められた OftpPsw が正確かどうかを確認します。IdToCheck と連携して使うことが必要です。

戻り値

このコマンドには、戻り値はありません。

関連トピック

OFTP プロトコル使用の詳細については、『コミュニケーションズ ゲートウェイ設定 ガイド』または『アドバンスド データ ディストリビューション ゲートウェイ設定 ガイド』の付録の「OFTP の操作」を参照してください。

OftpSetDynamicPassword

コマンド書式

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

コマンドの説明

このコマンドは、ホストがパスワードの変更をサポートしている場合に、ローカル マシンとホスト マシンの両方で OFTP パスワードを動的に更新するために使用します。このコマンドを正常に機能させるには、OftpRemote コマンドの最初の 3 個のパラメータに scriptvar 型の変数を使わなければなりません。このコマンドの 3 通りの形式を次の例に示します。

パラメータ

Switch	オプション。新しい OFTP パスワードに使う SSID のフィールドを示す整数値です。 既定値の [FALSE] に設定すると、新しいパスワードが SSID の ssiduser フィールドに送られ、古いパスワードは ssidpwsd フィールドに送られます。 [TRUE] に設定すると、新しいパスワードが SSID の ssidpwsd フィールドに送られ、古いパスワードは ssiduser フィールドに送られます。
PswExpire	オプション。パスワードが期限切れになるときを (日数単位で) 示す整数値です。このパラメータを使う場合は、Switch パラメータも指定しなければなりません。そのパスワードが期限切れになると、新しいパスワードがユーザーのために自動的に生成されます。

戻り値

このコマンドには、戻り値はありません。

(次のページへ続く)

例

```
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);
```

関連トピック

OFTP プロトコル使用の詳細については、『コミュニケーションズ ゲートウェイ 設定 ガイド』または『アドバンスド データ ディストリビューション ゲートウェイ設定 ガイド』の付録の「OFTP の操作」を参照してください。

OftpSetEerpDelivered

コマンド書式

```
OftpSetEerpDelivered();
```

コマンドの説明

送信されたファイル用に受信された EERP は、既定で、メールボックス システムで PICKEDUP とマークされます。OftpSetEerpDelivered () コマンドは、既定の設定を上書きし、EERP を DELIVERED とマークします。

パラメータ

このコマンドには、パラメータはありません。

戻り値

このコマンドには、戻り値はありません。

例

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

関連トピック

OFTP プロトコル使用の詳細については、『コミュニケーションズ ゲートウェイ設定 ガイド』または『アドバンスド データ ディストリビューション ゲートウェイ設定 ガイド』の付録の「OFTP の操作」を参照してください。

OftpSetMaxRecordSize

コマンド書式

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

コマンドの説明

このコマンドは、既定のレコード サイズの 0 (既定のレコード形式の "U" で使用) を上書きします。このコマンドは、多くの場合は OftpSetRecordFormat と連携して使用します。

パラメータ

RecordSize	レコード の形式設定で使う レコード サイズを指定します。

戻り値

このコマンドには、戻り値はありません。

例

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

関連トピック

詳細については、この章の「OftpSetRecordFormat」を参照してください。

OftpSetRcvDupCheck

コマンド書式

```
OftpSetRcvDupCheck();
```

コマンドの説明

OftpRemote や OftpHost セッションでファイルが受信されると、このコマンドは、SFID の sfiddsn、sfiddate、sfidtime、および sfidorig フィールドの値に基づいて、重複ファイルの有無を確認して存在すれば拒否します。

パラメータ

このコマンドには、パラメータはありません。

戻り値

このコマンドには、戻り値はありません。

関連トピック

OFTP プロトコル使用の詳細については、『コミュニケーションズ ゲートウェイ設定 ガイド』または『アドバンスド データ ディストリビューション ゲートウェイ設定 ガイド』の付録の「OFTP の操作」を参照してください。

OftpSetRecordFormat

コマンド書式

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

コマンドの説明

このコマンドは、既定のレコード形式 "U" を上書きします。"U" は、コミュニケーション セッション全体で使われます。レコード形式の上書きがメッセージ添付のコンテンツ タイプで指定されている場合、OftpSetRecordFormat コマンドが上書きされます。OftpSetRecordFormat は、多くの場合は **OftpSetMaxRecordSize** と連携して使用します。

パラメータ

RecordFormat	有効な値は、[F]、[V]、[U] および [T] です。

戻り値

このコマンドには、戻り値はありません。

例

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

関連トピック

詳細については、この章の「OftpSetMaxRecordSize」を参照してください。

OftpSetSpecialEERP

コマンド書式

```
OftpSetSpecialEERP();
```

コマンドの説明

このコマンドは、destination フィールドと originator フィールドをスワップする特殊な end-to-end 応答パケット処理を使うよう指定します。これは、通常、TRADANET タイプのネットワークで使われます。

パラメータ

このコマンドには、パラメータはありません。

戻り値

このコマンドには、戻り値はありません。

関連トピック

OFTP プロトコル使用の詳細については、『コミュニケーションズ ゲートウェイ設定 ガイド』または『アドバンスド データ ディストリビューション ゲートウェイ設定 ガイド』の付録の「OFTP の操作」を参照してください。

OftpSpecialLogicOff

コマンド書式

```
OftpSpecialLogicOff();
```

コマンドの説明

このコマンドは、ネゴシエーションで使う初期的な特殊論理設定をオフにするために使用します。これを使用するのは X.25 接続を使うときだけで、X.25 接続を行う前に使う必要があります。リモート サイトは、特殊な論理を使うためにネゴシエーションを行う場合があります。

パラメータ

このコマンドには、パラメータはありません。

戻り値

このコマンドには、戻り値はありません。

関連トピック

OFTP プロトコル使用の詳細については、『コミュニケーションズ ゲートウェイ設定 ガイド』または『アドバンスド データ ディストリビューション ゲートウェイ設定 ガイド』の付録の「OFTP の操作」を参照してください。

ParseNamed

コマンド書式

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

コマンドの説明

このコマンドは、バッファから " チャンク " を抽出するために、データ内で識別する " 名前 " を探します。現在の解析位置とは無関係に、バッファ全体が検索されます。これで、スクリプトは任意の順序で登場する指定されたパラメータを処理できます。チャンクがその抽出後に以前の解析位置を通り過ぎたら、現在の解析位置は順方向に送られます。つまり、**ParseNamed** を数回呼び出した後に **ParseNext** を呼び出した場合、**ParseNext** は文字列に順送りされた指定の " チャンク " の末尾から検索を始めます。

パラメータ

sName	データ内で一致する文字列値です。
sEndChars	検索の最終点を示すために使う一連の文字列です。
sChunk	データから取得する文字列値です。

戻り値

このコマンドには、戻り値はありません。

例

```
// 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

コマンド書式

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

コマンドの説明

このコマンドは、バッファから次の " チャンク " のデータを抽出します。これは、現在の解析位置から始めて、sSkipChars パラメータで示されていない文字を検出するまでスキップして、チャンクの先頭を見つけます。それから、sEndChars パラメータに記述された文字を見つけてチャンクの末尾を検出し、それに応じて現在の解析位置を設定します。

パラメータ

sSkipChars	バッファで無視される一連の文字列です。
sEndChars	検索の最終点を示すために使う一連の文字列です。
sChunk	データから取得する文字列値です。

戻り値

このコマンドには、戻り値はありません。

例

```
// 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

コマンド書式

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

コマンドの説明

このコマンドは、現在の解析位置を nSkip の数の文字を順送りします。

パラメータ

nSkip	スキップする文字数です。

戻り値

このコマンドには、戻り値はありません。

ParseStart

コマンド書式

```
ParseStart();
```

コマンドの説明

このコマンドは、各バッファを解析する前にリモート コンピュータから呼び出す必要があります。これは、現在の解析位置をバッファの先頭にリセットします。

パラメータ

このコマンドには、パラメータはありません。

戻り値

このコマンドには、戻り値はありません。

例

```
// 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

コマンド書式

```
Pause(integer Seconds);
```

コマンドの説明

このコマンドは、指定した秒数だけスクリプトの実行を一時停止するために使用します。

パラメータ

Seconds	一時停止する秒数が入ります。

戻り値

このコマンドには、戻り値はありません。

RcvBufSearch

コマンド書式

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

メッセージの説明

このコマンドは、最後の受信バッファで指定した文字列を検索するために使用します。このコマンドは、制御情報を受信して受信内容を確認するのに役立ちます。

パラメータ

SearchStr	最後の受信バッファで検索する文字列が入ります。

戻り値

0	文字列バッファに検索文字列が入っていない場合
1	文字列バッファに検索文字列が入っている場合

例

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

right

コマンド書式

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

コマンドの説明

right 関数は、文字列変数または文字列フィールド内の右側から指定した文字数を抽出します。

パラメータ

string_variable	操作する文字からなる文字列の名前です。
num_char	文字列の右側からカウントする文字数です。

戻り値

この関数は、文字列から一連の文字を返します。

例

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

scriptvar

コマンド書式

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

コマンドの説明

このコマンドは、ユーザー インターフェイスで編集用に使えるスクリプト変数を定義します。スクリプト変数は必須ではありませんが、使用した場合には、特定情報を処理するために使いやすいユーザー インターフェイスを提供します。スクリプト変数に意味のある名前を指定することで、ユーザーは指定する必要のある情報をすぐに判断できます。" オプション " パラメータを使ってスクリプト変数がオプションであることを指定済みでない限り、スクリプト変数を作成すると、ユーザーはその変数の値を入力するよう要求されます。

パラメータ

optional	このスクリプト変数をオプション変数として識別します。
type	変数の型です (整数、実数、文字列、日付 / 時刻、または配列)。
name	変数の名前です。

戻り値

このコマンドには、戻り値はありません。

例

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

set

コマンド書式

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

コマンドの説明

set 関数を使うと、日付 / 時刻変数の個々のコンポーネントを定義できます。

パラメータ

datetime_component	日付 / 時刻変数の個々のコンポーネントです。
datetime_variable	アクセスするコンポーネントの日付 / 時刻変数です。
integer_variable	整数変数です。

戻り値

このコマンドには、戻り値はありません。

例

```
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

コマンド書式

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

コマンドの説明

このコマンドは、各種プロトコルに使うブロック サイズをセットまたはリセットするために使用します。

パラメータ

BlockSize	転送で使うブロック サイズをセットまたはリセットする値が入ります。

戻り値

このコマンドには、戻り値はありません。

SetBufferSize

コマンド書式

```
SetBufferSize(integer BufferSize);
```

コマンドの説明

このコマンドは、内部読み書きバッファのサイズをリセットするために使用します。既定値は 2048 です。

パラメータ

BufferSize	内部読み書きバッファの新しいサイズが入ります。

戻り値

このコマンドには、戻り値はありません。

SetRcvError

コマンド書式

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

コマンドの説明

このコマンドは、特定プロトコルを使ったダウンロードでスキャンする文字列でエラー条件を示すように指定するために使用します。これは、エラー条件が発生したことを示す、データや 1 つの文字列のみを受信するかもしれない状況で役立ちます。

パラメータ

RcvErrorIndicator	受信エラー条件を示す文字列値です。

戻り値

このコマンドには、戻り値はありません。

SetRcvFileMode

コマンド書式

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

コマンドの説明

このコマンドは、システムがローカルに作成されたファイルに上書きや追加を行うかどうかを制御します。

パラメータ

Mode	有効な値は [APPEND] または [OVERWRITE] です。

戻り値

このコマンドには、戻り値はありません。

例

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

SetRcvNoData

コマンド書式

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

コマンドの説明

このコマンドは、特定プロトコルを使ったダウンロードでスキャンする文字列で受信できるデータがないことを示すように指定するために使用します。これは、受信できるデータがないことを示すデータや 1 つの文字列のみを受信するかもしれない状況で役立ちます。

パラメータ

RcvNoDataIndicator　受信データなしの条件を示す文字列値です。

戻り値

このコマンドには、戻り値はありません。

SetSessionType

コマンド書式

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

コマンドの説明

このコマンドは、システムが実行するコミュニケーション セッションの種類を上書きします。このコマンドを使うのは、システムが特定種類のセッションでコミュニケーションを開始できない場合です。

パラメータ

Type	有効な値は次のとおりです。 0 = 送信専用 1 = 送受信 2 = 受信専用

戻り値

このコマンドには、戻り値はありません。

例

```
SetSessionType(0);
```

SetSpecialUpdate

コマンド書式

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

説明

このコマンドは、コミュニケーション セッションの通常処理を上書きします。このコマンドが有効になると、スクリプト内の SndOk への呼出しが上書きされ、SetStatus (SUCCESS) コマンドが実行されるまで送信済みとマークされることはありません。また、SetStatus (SUCCESS) コマンドが実行されるまで、受信されたどのデータもメールボックス システムに送られません。このコマンドは、" オール オア ナッシング " 論理とも呼ばれる " セッション レベル " を使う特定ネットワークとコミュニケーションをとるときに役立ちます。

パラメータ

TrueOrFalse	このコマンドを有効または無効にするために使うブール フラグです。

戻り値

このコマンドには、戻り値はありません。

SetStatus

コマンド書式

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

コマンドの説明

このコマンドは、セッションの状態を設定します。定義済み定数値の [SUCCESS] と [FAILED] のどちらかを使用します。

パラメータ

Status	セッションの状態を設定する値 ([SUCCESS] または [FAILED]) が入ります。

戻り値

このコマンドには、戻り値はありません。

例

```
SetStatus(SUCCESS);
```

SetTimeout

コマンド書式

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

コマンドの説明

このコマンドは、コミュニケーション セッションで使う新しいタイムアウト値を設定します。

パラメータ

Seconds	秒単位の新しいタイムアウト値です (既定値は 60)。

戻り値

このコマンドには、戻り値はありません。

例

```
SetTimeout(60);
```

SndOK

コマンド書式

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

コマンドの説明

このコマンドは、メールボックス メッセージの正常な送信を示すために使用します。このコマンドはメッセージとそのすべての添付が正常に送信されるたびに必ず発行し、状態インジケータを更新して、メッセージが再び送信されないようにします。このコマンドは添付が正常に送信されたときにも発行し、状態インジケータを更新して、添付が再び送信されないようにします。

パラメータ

MsgId	状態を更新するためのメールボックス メッセージ識別子です。AtmId で指定された添付のメールボックス メッセージ識別子です。
AtmId	状態を更新するためのメールボックス添付識別子です。

戻り値

このコマンドには、戻り値はありません。

例

```
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

コマンド書式

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

コマンドの説明

strdate 関数は、指定した書式で日付 / 時刻型を文字列に変換します。この関数ではスラッシュ (/) などの静的文字を含めることができるため、日付のサポートをすべて利用できます。

パラメータ

datetime　日付 / 時刻変数です (月は 0 ～ 11 で指定します)。

format　目的の日付書式です。書式指定子は次のとおりです。

書式指定子	説明
%a	略式曜日名
%A	完全曜日名
%b	略式月名
%B	完全月名
%d	10 進数での日付 (01 ～ 31)
%H	24 時間形式の時間 (00 ～ 23)
%I	12 時間形式の時間 (01 ～ 12)
%j	10 進数での年間通算日 (001 ～ 366)
%m	10 進数での月 (01 ～ 12)
%M	10 進数での分 (00 ～ 59)
%S	10 進数での秒 (00 ～ 59)
%U	日曜日を週初日とした、10 進数での年間通算週 (00 ～ 51)
%w	10 進数での曜日 (0 ～ 6、日曜日を "0" として)

(次のページへ続く)

（続き） 書式指定子	説明
%W	月曜日を週初日とした、10 進数での年間通算週 (00 ～ 51)
%y	10 進数での 2 桁年数 (00 ～ 99)
%Y	10 進数での 4 桁年数
%%	パーセント記号

string　　文字列変数

戻り値

このコマンドには、戻り値はありません。

例

```
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

コマンド書式

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

コマンドの説明

strstr 関数は、文字列内の部分文字列を検索します。

パラメータ

integer	整数変数です。
string	評価する文字列です。
substring	検索対象の文字列の一部です。

戻り値

この関数は、指定の文字列内の指定した部分文字列の最初のインスタンスの文字位置を返します。ただし、部分文字列が見つからない場合には -1 を返します。

例

```
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

コマンド書式

```
TipRemote();
```

説明

このコマンドは、Tradanet Interface Protocol (TIP) を使って Tradanet ネットワークに対して完全なリモート セッションを実行するために使用します。このコマンドを使うには、まずメールボックス構成プロパティで Tradanet コマンドを有効にして、TIP コマンドを選択してから該当するオプションを構成します。

戻り値

このコマンドには、戻り値はありません。

例

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

TipSetFwd

コマンド書式

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

説明

このコマンドは、TipRemote セッションで使うブロック転送特性を設定します。

パラメータ

Forwarding	0 — 転送文字がありません。 1 — 入力サービスで転送文字があります。 2 — 出力サービスで転送文字があります。 3 — (既定値) 入出力の両方に転送文字があります。

戻り値

このコマンドには、戻り値はありません。

例

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

TipSetPadding

コマンド書式

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

説明

このコマンドは、TipRemote セッションで使うブロック付け足し特性を設定します。

パラメータ

Padding

N — (既定値) 付け足しなしです。すべてのブロックで物理的なサイズが使用されます。

Y — データ ブロックに付け足します。データの最終ブロックに付け足して blocksize にします。

A — すべてのブロック (擬似命令およびデータ) に付け足して blocksize にします。

戻り値

このコマンドには、戻り値はありません。

例

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

TipSetPadChar

コマンド書式

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

説明

このコマンドは、TipRemote セッション中にブロック付け足しに使う付け足し文字を設定します。

パラメータ

PadChar	既定値は 0 です。blocksize にするためにブロックに付け足す文字です。

戻り値

このコマンドには、戻り値はありません。

例

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

TipSetConType

コマンド書式

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

説明

このコマンドは、TipRemote セッションで使う接続の種類を設定します。

パラメータ

ConType	1 — LAN
	2 — 既定値 X25
	3 — その他

戻り値

このコマンドには、戻り値はありません。

例

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

While...Do

コマンド書式

```
while condition do
```

コマンドの説明

while...do キーワードは、指定した終了条件がゼロに等しくなるまでステートメントを繰り返し実行します。システムはループの各繰返しが実行される前に終了条件をテストするため、**while** ループは終了式の値に応じて 0 回以上実行されます。

パラメータ

このコマンドには、パラメータはありません。

戻り値

このコマンドには、戻り値はありません。

例

```
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

コマンド書式

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

コマンドの説明

winexec 関数を使うと、トランスレータを実行しながら別のプログラムを実行できます。このプログラムは非同期で実行されます。プログラムを指定して、プログラム ウィンドウの表示方法を決めます。必要な場合は、エラー コードを返すこともできます。エラー コードが 32 より大きい場合は、プログラムはエラーなしで実行されています。エラー コードが 32 未満の場合は、エラーのためにプログラムは実行されていません。エラー コードが "0" の場合はメモリ不足です。エラー コードが "2" の場合は、ファイル名が指定されていません。エラー コードは、実行されたプログラムからの戻り値ではありません。

パラメータ

program　実行可能プログラム名の文字列です。

window_display　プログラム ウィンドウの表示方法を示す番号です。次の表では、プログラム ウィンドウの外観を制御する window_display 番号を示しています。window_display 値ではなく次の番号を使うと、プログラム ウィンドウの表示方法を示すことができます。

番号	Window_Display	定義
0	SW_HIDE	そのウィンドウを表示せずに別のウィンドウをアクティブにします。
1	SW_SHOWNORMAL	ウィンドウをアクティブにして表示します。ウィンドウが最小化または最大化されている場合は、元のサイズおよび位置に戻されます。ウィンドウを初めて表示する場合は、このフラグを指定する必要があります。
1	SW_NORMAL	ウィンドウをアクティブにして、元のサイズおよび位置で表示します。
2	SW_SHOWMINIMIZED	ウィンドウをアクティブにし、最小化ウィンドウとして表示します。

(次のページへ続く)

(続き) 番号	Window_Display	定義
3	SW_SHOWMAXIMIZED	ウィンドウをアクティブにし、最大化ウィンドウとして表示します。
3	SW_MAXIMIZE	ウィンドウを最大化します。
4	SW_SHOWNOACTIVATE	最後に使用したサイズおよび位置でウィンドウを表示しますが、アクティブにはしません (現在のアクティブ ウィンドウがアクティブのままになります)。
5	SW_SHOW	ウィンドウをアクティブにして、現在のサイズおよび位置で表示します。
6	SW_MINIMIZE	ウィンドウを最小化します。
7	SW_SHOWMINNOACTIVE	ウィンドウをアクティブにせずに最小化ウィンドウとして表示します (現在のアクティブ ウィンドウがアクティブのままになります)。
8	SW_SHOWNA	ウィンドウをアクティブにせずに最後に使用したサイズおよび位置で表示します (現在のアクティブ ウィンドウがアクティブのままになります)。
9	SW_RESTORE	ウィンドウをアクティブにして表示します。ウィンドウが最小化または最大化されている場合は、元のサイズおよび位置に戻されます。
10	SW_SHOWDEFAULT	ウィンドウをアクティブにすると、Windows によってサイズおよび位置が自動調整されます。

戻り値

この関数は、オペレーティング システムが提供する結果値を返します。

(次のページへ続く)

例

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

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

WwaLogoff

コマンド書式

```
WwaLogoff();
```

コマンドの説明

このコマンドは、World Wide Async (WWA) プロトコルを使ってログオフ手順を実行します。

パラメータ

このコマンドには、パラメータはありません。

戻り値

このコマンドには、戻り値はありません。

WwaLogon

コマンド書式

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

コマンドの説明

このコマンドは、World Wide Async プロトコルを使ってログオン手順を実行します。

パラメータ

LogonAcct	ログオン アカウントが入ります。
LogonId	ログオン識別子が入ります。
LogonPsw	ログオン パスワードが入ります。
IEAcct	情報交換アカウントが入ります。
IEId	情報交換識別子が入ります。
IEPsw	情報交換パスワードが入ります。
NewLogonPsw	新しいログオン パスワードが入ります。これを使うのは、ログオン パスワードを変更する場合のみです。
NewIEPsw	新しい情報交換パスワードが入ります。これを使うのは、情報交換パスワードを変更する場合のみです。

戻り値

このコマンドには、戻り値はありません。

WwaRcvMsg

コマンド書式

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

コマンドの説明

このコマンドは、World Wide Async プロトコルを使ってデータを受信するために使用します。受信データは、1 つの添付と一緒にメールボックス メッセージに入れられます。

パラメータ

UserMsgClass	オプション。これを指定すると、指定されたユーザー メッセージ分類を保持するデータを取得します。

戻り値

このコマンドには、戻り値はありません。

WwaSndAtm

コマンド書式

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

コマンドの説明

このコマンドは、World Wide Async プロトコルを使ってメールボックス添付を送信するために使用します。

パラメータ

MsgId	**MbxGetNextMsg** から返されたメールボックス メッセージ識別子が入ります。
AtmId	**MbxGetNextAtm** から返されたメールボックス添付識別子が入ります。

戻り値

このコマンドには、戻り値はありません。

例

```
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

コマンド書式

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

コマンドの説明

このコマンドは、Xmodem プロトコルを使ってファイルを受信するために使用します。SetRcvFileMode コマンドは、システムがファイルに上書きや追加を行うかどうかを制御します。既定では、システムはファイルを上書きします。

パラメータ

Filename	データの受信に使うファイルの名前です。UNC (汎用命名規則) の名前は有効です。

戻り値

このコマンドには、戻り値はありません。

例

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

XmodemRcvMsg

コマンド書式

```
XmodemRcvMsg();
```

コマンドの説明

このコマンドは、Xmodem プロトコルを使ってデータを受信するために使用します。受信データは、1 つの添付と一緒にメールボックス メッセージに入れられます。

パラメータ

このコマンドには、パラメータはありません。

戻り値

このコマンドには、戻り値はありません。

XmodemSetFillChar

コマンド書式

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

コマンドの説明

このコマンドは、最終ブロック全体にデータが入っていない場合にそのブロックに付け足すように、Xmodem プロトコルで使う充填文字を設定します。既定の充填文字は 0 (NULL) です。

パラメータ

FillChar	必要に応じて、最終データ ブロックの末尾に充填するのに使う文字を指定します。下記の例のように、^xx という形式の 16 進数値で指定できます。

戻り値

このコマンドには、戻り値はありません。

例

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

XmodemSndAll

コマンド書式

```
XmodemSndAll();
```

コマンドの説明

このコマンドは、Xmodem プロトコルを使ってすべての使用可能なメッセージ添付を 1 つのファイルで送信するために使用します。このコマンドは内部的に特定のタスクを実行するため、次の必要がありません。

- 各メッセージと添付のためにループを定義する
- SndOk スクリプト コマンドを使用する

パラメータ

このコマンドには、パラメータはありません。

戻り値

このコマンドには、戻り値はありません。

例

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

XmodemSndAtm

コマンド書式

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

コマンドの説明

このコマンドは、Xmodem プロトコルを使ってメールボックス添付を送信するために使用します。

パラメータ

MsgId	**MbxGetNextMsg** から返されたメールボックス メッセージ識別子が入ります。
AtmId	**MbxGetNextAtm** から返されたメールボックス添付識別子が入ります。

戻り値

このコマンドには、戻り値はありません。

例

```
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

コマンド書式

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

コマンドの説明

このコマンドは、Xmodem プロトコルを使ってファイルを送信するために使用します。

パラメータ

Filename	送信するファイルの名前です。UNC 名を適用できます。

戻り値

このコマンドには、戻り値はありません。

例

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

ZmodemRcvFile

コマンド書式

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

コマンドの説明

このコマンドは、システムに対して、1 つまたは複数の Zmodem プロトコルを使うファイルを受信する方法を指示します。受信されたファイルは、既定で既存ファイルを上書きします。

SetRcvFileMode コマンドと連携して使うと、受信された情報を既存ファイルに付加するようシステムに指示できます。

パラメータ

FilenameOrPath	固定のファイル名またはパスを指定します。固定ファイル名を使う場合、すべての受信済みファイルはこの 1 つのファイルに書き込まれます。パス名を指定すると、受信される各ファイルについてその位置に個別のファイルが作成されます。UNC (汎用命名規則) の名前は有効なエントリです。
PathIndicator	オプション。既定の設定は FALSE です。TRUE に設定すると、パス名パラメータを指定しなければなりません。

戻り値

このコマンドには、戻り値はありません。

例 1

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

例 2

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

ZmodemRcvMsg

コマンド書式

```
ZmodemRcvMsg();
```

コマンドの説明

このコマンドは、Zmodem プロトコルを使ってデータを受信するために使用します。

パラメータ

このコマンドには、パラメータはありません。

戻り値

このコマンドには、戻り値はありません。

例

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

ZmodemSndAll

コマンド書式

```
ZmodemSndAll();
```

コマンドの説明

このコマンドは、Zmodem プロトコルを使ってすべてのファイルを送信するために使用します。

パラメータ

このコマンドには、パラメータはありません。

戻り値

このコマンドには、戻り値はありません。

例

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

ZmodemSndFile

コマンド書式

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

コマンドの説明

このコマンドは、Zmodem プロトコルを使ってファイルを送信するために使用します。

パラメータ

Filename	送信するファイルの名前です。UNC 名を適用できます。

戻り値

このコマンドには、戻り値はありません。

例

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