HITACHI
Inspire the Next

JP1 Version 9

JP1/NETM/DM Automatic Installation Tool ガイド (Windows(R) 用)

文法・操作書

3020-3-S83-80

■対象製品

P-2642-1194 JP1/NETM/DM Manager 09-51 （適用 OS：Windows Server 2003，Windows XP Professional，Windows 2000）＊
P-2642-1394 JP1/NETM/DM Client 09-51 （適用 OS：Windows Server 2003，Windows XP，Windows 2000，Windows NT 4.0，Windows Me，Windows 98）＊
P-2642-2394 JP1/NETM/DM Client - Base 09-51 （適用 OS：Windows Server 2003，Windows XP，Windows 2000，Windows NT 4.0，Windows Me，Windows 98）＊
P-F2642-23941 JP1/NETM/DM Client - Operation Log Feature 09-51 （適用 OS：Windows Server 2003，Windows XP，Windows 2000，Windows NT 4.0，Windows Me，Windows 98）＊
P-F2642-23942 JP1/NETM/DM Client - Delivery Feature 09-51 （適用 OS：Windows Server 2003，Windows XP，Windows 2000，Windows NT 4.0，Windows Me，Windows 98）＊
P-F2642-23943 JP1/NETM/DM Client - Remote Control Feature 09-51 （適用 OS：Windows Server 2003，Windows XP，Windows 2000，Windows NT 4.0，Windows Me，Windows 98）＊
P-2A42-1194 JP1/NETM/DM Manager 09-51 （適用 OS：Windows 8，Windows Server 2012，Windows 7，Windows Server 2008，Windows Vista）＊
P-2C42-1394 JP1/NETM/DM Client 09-51 （適用 OS：Windows 8，Windows Server 2012，Windows 7，Windows Server 2008，Windows Vista）＊
P-2C42-2394 JP1/NETM/DM Client - Base 09-51 （適用 OS：Windows 8，Windows Server 2012，Windows 7，Windows Server 2008，Windows Vista）＊
P-F2C42-23941 JP1/NETM/DM Client - Operation Log Feature 09-51 （適用 OS：Windows 8，Windows Server 2012，Windows 7，Windows Server 2008，Windows Vista）＊
P-F2C42-23942 JP1/NETM/DM Client - Delivery Feature 09-51 （適用 OS：Windows 8，Windows Server 2012，Windows 7，Windows Server 2008，Windows Vista）＊
P-F2C42-23943 JP1/NETM/DM Client - Remote Control Feature 09-51 （適用 OS：Windows 8，Windows Server 2012，Windows 7，Windows Server 2008，Windows Vista）＊
＊印の製品は，ISO9001 および TickIT の認証を受けた品質マネジメントシステムで開発されました。

■輸出時の注意

本製品を輸出される場合には、外国為替及び外国貿易法の規制並びに米国輸出管理規則など外国の輸出関連法規をご確認の上、必要な手続きをお取りください。
なお、不明な場合は、弊社担当営業にお問い合わせください。

■商標類

■マイクロソフト製品のスクリーンショットの使用について

Microsoft Corporation のガイドラインに従って画面写真を使用しています。

■発行

2013 年 2 月　3020-3-S83-80

■著作権

変更内容

変更内容（3020-3-S83-80） JP1/NETM/DM 09-51

追加・変更内容	変更個所
適用 OS に Windows 8 および Windows Server 2012 を追加した。	1.3，2.1.2，2.4.1，2.6.1，2.7，4.2

単なる誤字・脱字などはお断りなく訂正しました。

はじめに

このマニュアルは，JP1/NETM/DM のコンポーネントである Automatic Installation Tool の使用方法の詳細を説明したものです。ソフトウェアの自動インストールに使用する AIT（Automatic Installation Tool）ファイルやレコーダファイルを作成する際にお読みください。

なお，このマニュアルを含め，Windows 版 JP1/NETM/DM のマニュアルには次の 7 冊があります。各マニュアルの目的を次に示しますので，必要に応じてお読みください。

JP1 Version 9 JP1/NETM/DM **導入・設計ガイド** (Windows(R) **用**)
: 最初にお読みいただくマニュアルです。
JP1/NETM/DM の概要や機能，代表的な構築例および使用例を紹介しています。また，JP1/NETM/DM を導入するための手順や，あらかじめ検討しておく必要があることについても説明しています。

JP1 Version 9 JP1/NETM/DM **構築ガイド** (Windows(R) **用**)
: JP1/NETM/DM のインストール・セットアップの手順，データベースの構築，およびシステム構成の管理方法について説明しています。

JP1 Version 9 JP1/NETM/DM **運用ガイド** 1(Windows(R) **用**)
: ソフトウェアの配布，インベントリ情報の取得および管理，ファイルの収集など，配布管理システムの各機能の詳細と操作方法を説明しています。また，クライアントの操作方法についても説明しています。

JP1 Version 9 JP1/NETM/DM **運用ガイド** 2(Windows(R) **用**)
: 他の製品と JP1/NETM/DM との連携，およびトラブルが発生したときの対処方法について説明しています。また，Windows 8・Windows Server 2012・Windows 7・Windows Server 2008・Windows Vista 版 JP1/NETM/DM Client，64 ビット版 JP1/NETM/DM Client，および Windows CE 版 JP1/NETM/DM Client の機能についても説明しています。

JP1 Version 9 JP1/NETM/DM Automatic Installation Tool **ガイド** (Windows(R) **用**)
: 他社ソフトウェアをパッケージングするときに使用する AIT ファイルやレコーダファイルの作成方法を説明しています。

JP1 Version 9 JP1/NETM/DM Administrator Kit
: JP1/NETM/DM Client を自動的にインストールするために使用する JP1/NETM/DM Administrator Kit について説明しています。

JP1 Version 9 JP1/NETM/Remote Control
: JP1/NETM/Remote Control および JP1/NETM/DM のリモートコントロール機能について説明しています。

■対象読者

このマニュアルは，次の方にお読みいただくことを前提に説明しています。

- JP1/NETM/DM を利用してソフトウェアの配布や資産情報を収集・管理する管理者の方
- Microsoft Windows の操作に関する基本的な知識をお持ちの方

■関連マニュアル

このマニュアルの関連マニュアルを次に示します。必要に応じてお読みください。

- JP1 Version 9 JP1/NETM/DM 導入・設計ガイド (Windows(R) 用)（3020-3-S79）
- JP1 Version 9 JP1/NETM/DM 構築ガイド (Windows(R) 用)（3020-3-S80）
- JP1 Version 9 JP1/NETM/DM 運用ガイド 1(Windows(R) 用)（3020-3-S81）
- JP1 Version 9 JP1/NETM/DM 運用ガイド 2(Windows(R) 用)（3020-3-S82）

- JP1 Version 9 JP1/NETM/DM Administrator Kit（3020-3-S84）
- JP1 Version 9 JP1/NETM/Remote Control（3020-3-S87）

■このマニュアルでの表記

このマニュアルでは，JP1/NETM/DM 関連製品の名称を次のように表記します。

表記	製品名称	
64 ビット版 JP1/NETM/DM Client	Windows Server 2003 (IPF) 対応 JP1/NETM/DM Client	
	Windows Server 2003 (IPF) 対応 JP1/NETM/DM Client Light Edition	
JP1/NETM/DM	JP1/NETM/DM Client	JP1/NETM/DM Client
		JP1/NETM/DM Client - Base
	JP1/NETM/DM Client - Delivery Feature	
	JP1/NETM/DM Client - Operation Log Feature	
	JP1/NETM/DM Client - Remote Control Feature	
	JP1/NETM/DM Manager	
Windows 8・Windows Server 2012・Windows 7・Windows Server 2008・Windows Vista 版 JP1/NETM/DM Client	Windows 8，Windows Server 2012，Windows 7，Windows Server 2008，および Windows Vista 対応の JP1/NETM/DM Client	

そのほかの製品名称，および名称について次のように表記します。

表記				製品名称および名称
ActiveX				ActiveX(R)
InstallShield				InstallShield(R)
Itanium 2				Intel(R) Itanium(R) 2 プロセッサ
Microsoft Internet Explorer				Microsoft(R) Internet Explorer
				Windows(R) Internet Explorer(R)
MS-DOS				Microsoft(R) MS-DOS(R)
Visual Test				Visual Test 4.0
				Visual Test 6.0
Windows	Windows 98			Microsoft(R) Windows(R) 98 Operating System
	Windows Me			Microsoft(R) Windows(R) Millennium Edition Operating System
	Windows NT	Windows 2000	Windows 2000 Advanced Server	Microsoft(R) Windows(R) 2000 Advanced Server Operating System
			Windows 2000 Datacenter Server	Microsoft(R) Windows(R) 2000 Datacenter Server Operating System
			Windows 2000 Professional	Microsoft(R) Windows(R) 2000 Professional Operating System
			Windows 2000 Server	Microsoft(R) Windows(R) 2000 Server Operating System

表記				製品名称および名称
		Windows 7		Microsoft(R) Windows(R) 7 Enterprise
				Microsoft(R) Windows(R) 7 Professional
				Microsoft(R) Windows(R) 7 Ultimate
		Windows 8		Microsoft(R) Windows(R) 8
				Microsoft(R) Windows(R) 8 Enterprise
				Microsoft(R) Windows(R) 8 Pro
		Windows NT 4.0	Windows NT Server 4.0	Microsoft(R) Windows NT(R) Server Network Operating System Version4.0
			Windows NT Workstation 4.0	Microsoft(R) Windows NT(R) Workstation Operating System Version4.0
		Windows Server 2003 ※1	Windows Server 2003 ※1	Microsoft(R) Windows Server(R) 2003 R2, Datacenter Edition
				Microsoft(R) Windows Server(R) 2003 R2, Enterprise Edition
				Microsoft(R) Windows Server(R) 2003 R2, Standard Edition
				Microsoft(R) Windows Server(R) 2003, Datacenter Edition
				Microsoft(R) Windows Server(R) 2003, Enterprise Edition
				Microsoft(R) Windows Server(R) 2003, Standard Edition
			Windows Server 2003 (IPF)	Microsoft(R) Windows Server(R) 2003, Datacenter Edition for Itanium-based Systems
				Microsoft(R) Windows Server(R) 2003, Enterprise Edition for Itanium-based Systems
			Windows Server 2003 (x64)	Microsoft(R) Windows Server(R) 2003 R2, Datacenter x64 Edition
				Microsoft(R) Windows Server(R) 2003 R2, Enterprise x64 Edition
				Microsoft(R) Windows Server(R) 2003 R2, Standard x64 Edition
				Microsoft(R) Windows Server(R) 2003, Datacenter x64 Edition
				Microsoft(R) Windows Server(R) 2003, Enterprise x64 Edition
				Microsoft(R) Windows Server(R) 2003, Standard x64 Edition
		Windows Server 2008 ※2	Windows Server 2008 ※2	Microsoft(R) Windows Server(R) 2008 Datacenter
				Microsoft(R) Windows Server(R) 2008 Datacenter without Hyper-V(R)

表記				製品名称および名称
				Microsoft(R) Windows Server(R) 2008 Enterprise
				Microsoft(R) Windows Server(R) 2008 Enterprise without Hyper-V(R)
				Microsoft(R) Windows Server(R) 2008 Standard
				Microsoft(R) Windows Server(R) 2008 Standard without Hyper-V(R)
			Windows Server 2008 R2	Microsoft(R) Windows Server(R) 2008 R2 Datacenter
				Microsoft(R) Windows Server(R) 2008 R2 Enterprise
				Microsoft(R) Windows Server(R) 2008 R2 Standard
		Windows Server 2012		Microsoft(R) Windows Server(R) 2012 Datacenter
				Microsoft(R) Windows Server(R) 2012 Standard
		Windows Vista		Microsoft(R) Windows Vista(R) Business
				Microsoft(R) Windows Vista(R) Enterprise
				Microsoft(R) Windows Vista(R) Ultimate
		Windows XP	Windows XP Home Edition	Microsoft(R) Windows(R) XP Home Edition Operating System
			Windows XP Professional	Microsoft(R) Windows(R) XP Professional Operating System

注※ 1

Windows Server 2003 (IPF) または Windows Server 2003 (x64) を併記している場合は，Windows Server 2003 に Windows Server 2003 (IPF) および Windows Server 2003 (x64) は含みません。

注※ 2

Windows Server 2008 R2 を併記している場合は，Windows Server 2008 に Windows Server 2008 R2 は含みません。

■このマニュアルで使用している英略語

このマニュアルで使用している主な英略語を次に示します。

英略語	正式名称
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
CD	Compact Disc
CD-ROM	Compact Disc Read Only Memory
DLL	Dynamic Linking Library
FD	Floppy Disk
GUI	Graphical User Interface
HD	Hard Disk
HTML	Hyper Text Markup Language
I/O	Input/Output
ID	Identifier
IME	Input Method Editor

英略語	正式名称
IP	Internet Protocol
IPF	Itanium(R) Processor Family
MBCS	Multi-Byte Character Set
MS-DOS	Microsoft Disk Operating System
ODBC	Open DataBase Connectivity
OS	Operating System
PC	Personal Computer
PDF	Portable Document Format
PP	Program Product

■マニュアル間の参照指示について

マニュアル「JP1/NETM/DM 導入・設計ガイド (Windows(R) 用)」,「JP1/NETM/DM 構築ガイド (Windows(R) 用)」,「JP1/NETM/DM 運用ガイド 1(Windows(R) 用)」,「JP1/NETM/DM 運用ガイド 2(Windows(R) 用)」または「JP1/NETM/DM Automatic Installation Tool ガイド (Windows(R) 用)」間で，相互にマニュアルを参照していただく場合，次の形式で参照指示しています。

『AA については，マニュアル「BBB」の「n.n.n　XXXXX」を参照してください。』

AA

参照していただく項目です。

BBB

参照先マニュアルの略称です。マニュアル名称の共通部分（「JP1/NETM/DM」および「(Windows(R) 用)」の部分）を省略しています。省略されている部分を補ってお読みください。

n.n.n

参照先の章・節・項番号です。(1) や (a) などの括弧付き項番が付く場合もあります。

XXXXX

参照先の標題（見出し）です。

■マニュアルで使用している記号

このマニュアルで使用している記号を次のように定義します。

記号	意味
［ ］	ウィンドウ，ダイアログボックス，タブ，パネル，メニュー，ボタン，アイコン，グループ，フォルダ，およびキーの名称を示す。
「 」	画面上の文字列，記号，およびジョブの名称を示す。
［A］－［B］	メニューを連続して選択することを示す。 （例）［ファイル］－［新規作成］ 上記の例では，［ファイル］メニューを選択して，プルダウンメニューから［新規作成］を選択することを示す。
［X］＋［Y］	キーを同時に押すことを示す。 （例）［Ctrl］＋［C］キー 上記の例では，［Ctrl］キーと［C］キーを同時に押すことを示す。
斜体文字	可変の値を示す。 （例） 日付は *YYYYMMDD* の形式で指定する。

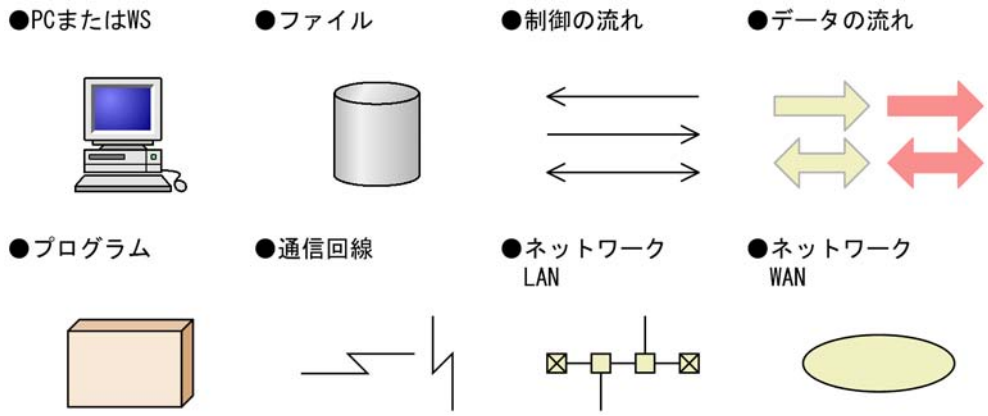
■文法で使用している記号

文法で使用している記号を次のように定義します。

記号	意味
| (ストローク)	複数の項目に対し，項目間の区切りを示し，「または」の意味を示す。 (例) A|B|C は，「A，B または C」を示す。
{ } (波括弧)	この記号で囲まれている複数の項目の中から，必ず 1 組の項目を選択する。項目の区切りは | で示す。 (例) {A|B|C} は「A，B または C のどれかを指定する」ことを示す。
[ ] (角括弧)	この記号で囲まれている項目は任意に指定できる (省略してもよい)。複数の項目が記述されている場合には，すべてを省略するか，どれか一つを選択する。 (例) [A] は「何も指定しない」か「A を指定する」ことを示す。[B|C] は「何も指定しない」か「B または C を指定する」ことを示す。
… (点線)	この記号の直前に示された項目を繰り返して複数個，指定できる。項目と項目の間は，一つ以上のスペースで区切る。 (例) A… は「A のあとに A を必要個数指定できる」ことを示す。
_ (下線)	括弧内のすべてを省略したときに，システムが採る標準値を示す。標準値がない場合は，指定した項目だけが有効である。 (例) [A|B] はこの項目を指定しなかった場合に，A を選択したと見なすことを示す。

■図中で使用している記号

このマニュアルの図中で使用している記号を，次のように定義します。



■ HTML ヘルプについて

JP1/NETM/DM では，次に示す HTML ヘルプを提供しています。

JP1/NETM/DM のヘルプ（JP1/NETM/DM Manager，JP1/NETM/DM Client（中継システム），および JP1/NETM/DM Client - Base（中継システム）用）

JP1/NETM/DM のヘルプの内容は，次のマニュアルを統合したものです。

・JP1 Version 9 JP1/NETM/DM 導入・設計ガイド (Windows(R) 用)
・JP1 Version 9 JP1/NETM/DM 構築ガイド (Windows(R) 用)
・JP1 Version 9 JP1/NETM/DM 運用ガイド 1(Windows(R) 用)
・JP1 Version 9 JP1/NETM/DM 運用ガイド 2(Windows(R) 用)
・JP1 Version 9 JP1/NETM/DM Automatic Installation Tool ガイド (Windows(R) 用)

JP1/NETM/DM Client のヘルプ（JP1/NETM/DM Client（クライアント）および JP1/NETM/DM Client - Base（クライアント）用）

JP1/NETM/DM Client のヘルプの内容は，上記のマニュアルからクライアントの説明を抜粋したものです。

これらの HTML ヘルプでは，検索したい項目を HTML ヘルプの全文から検索できます。
JP1/NETM/DM の各ウィンドウの［ヘルプ］メニューや各ダイアログボックスの［ヘルプ］ボタンから，HTML ヘルプを起動できます。HTML ヘルプは，Microsoft Internet Explorer 5.01 以降がインストールされている PC で参照してください。

■ KB（キロバイト）などの単位表記について

1KB（キロバイト），1MB（メガバイト），1GB（ギガバイト），1TB（テラバイト）はそれぞれ 1,024 バイト，$1{,}024^2$ バイト，$1{,}024^3$ バイト，$1{,}024^4$ バイトです。

目次

1 AIT ファイルを使ったリモートインストール

JP1/NETM/DM では，ソフトウェアをリモートインストールするとき，クライアントユーザがインストーラに応答することなく，自動インストールすることができます。この章では，自動インストールのために必要な AIT ファイルと，AIT ファイルを使ったリモートインストールの方法について説明します。また，AIT ファイルを作成・使用する場合の注意事項について説明します。

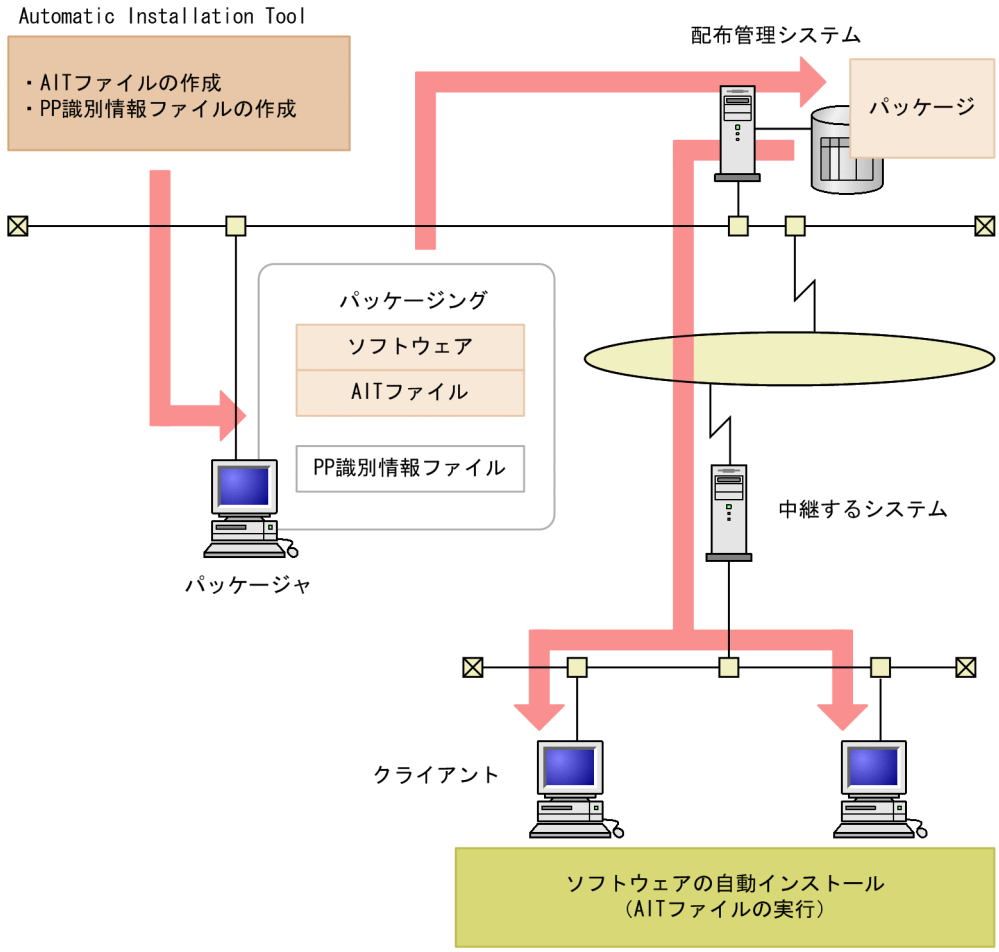
1.1　AIT ファイルとは

AIT **ファイル**とは，ソフトウェアのインストーラに自動応答する内容を記述したスクリプトファイルです。配布するソフトウェアとともに AIT ファイルをパッケージングし，リモートインストールすると，クライアントユーザがインストーラに応答することなく，ソフトウェアを自動インストールできます。

他社ソフトウェアや，インストーラに応答する必要のあるユーザ作成のプログラムをリモートインストールする場合は，AIT ファイルを作成してください。AIT ファイルは，JP1/NETM/DM のコンポーネントである Automatic Installation Tool を使用して作成します。

AIT ファイルを使ったリモートインストールの概要を，次の図に示します。

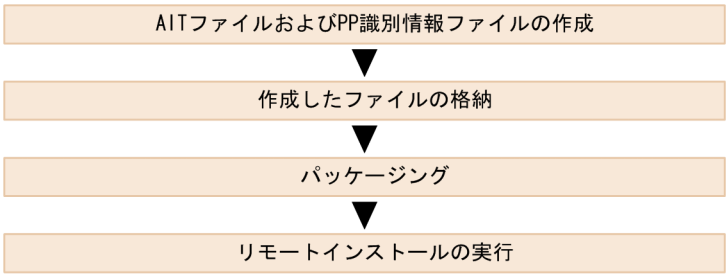
図 1-1　AIT **ファイルを使ったリモートインストール**



1.2　AIT ファイルを使ったリモートインストール手順

AIT ファイルを使ったリモートインストールの手順を次の図に示します。

図 1-2　AIT ファイルを使ったリモートインストール手順



1.2.1　AIT ファイルおよび PP 識別情報ファイルの作成

配布するソフトウェアのインストール手順を調査して，インストーラに自動応答する AIT ファイルを作成します。AIT ファイルは，Automatic Installation Tool 独自の AIT 言語で作成します。

また，AIT ファイルの作成時に PP **識別情報ファイル**を作成します。PP 識別情報ファイルには，配布するソフトウェアを AIT ファイルに関連づけるための情報が記述されています。PP 識別情報ファイルは，PPDEFAIT.DMP という名称で，決められた場所に格納しておく必要があります。

なお，PP 識別情報ファイルには，次の 2 種類があります。

● 標準添付の PP 識別情報ファイル
JP1/NETM/DM が提供する AIT ファイルに対応する，標準添付の PP 識別情報ファイルです。変更はしないでください。標準添付の PP 識別情報ファイルは，JP1/NETM/DM のインストール先フォルダ ¥MASTER 下に，PPDEFAIT.DMP というファイル名で格納されています。

● ユーザ作成の PP 識別情報ファイル
ユーザが作成した AIT ファイルを配布するソフトウェアと関連づけるために，ユーザが作成する PP 識別情報ファイルです。

AIT ファイルおよび PP 識別情報ファイルの作成方法については，「2.　AIT ファイルの作成」以降を参照してください。

また，配布するソフトウェアが Windows Installer を使用している場合の AIT ファイルの作成方法については，「付録 E　Windows Installer に対応したソフトウェアのリモートインストール手順」を参照してください。

なお，JP1/NETM/DM では，主なソフトウェアに対応する AIT ファイルを標準添付しています。標準添付している AIT ファイルについては，「付録 B　JP1/NETM/DM で提供する AIT ファイル」を参照してください。

1.2.2　作成したファイルの格納

ユーザが作成した PP 識別情報ファイルは，パッケージャの PC で，次のディレクトリに格納してください。

パッケージャのインストール先ディレクトリ `¥DMPRM¥PPDEFAIT.DMP`

また，ユーザが作成した AIT ファイルは，パッケージャの PC の，PP 識別情報ファイルの設定時に指定したパスに格納してください。

1.2.3 パッケージング

AIT ファイルと PP 識別情報ファイルを所定の場所に格納したあと，配布するソフトウェアをパッケージャでパッケージングします。

パッケージングする際，［JP1/NETM/DM パッケージング］ダイアログボックスの［パッケージング情報］パネルには，AIT ファイルおよび PP 識別情報ファイルで定義した「パッケージ識別 ID」，「バージョン」，「パッケージ名」が表示されます。この値は変更できません。

図 1-3 ［パッケージング情報］パネル

［JP1/NETM/DM パッケージング］ダイアログボックスには，AIT ファイルで定義した次の情報も表示されます。この値は，パッケージング時またはリモートインストール時に変更できます。

- インストール先ディレクトリ
- 会社名
- 所有者名
- シリアルナンバー
- アイコングループ

インストール中にクライアントユーザがダイアログボックスに応答したり，キーボードやマウスを操作したりすると，インストーラの画面表示が AIT ファイルの期待するものと異なるものになり，AIT ファイルによるリモートインストールが進まなくなる場合があります。このような場合に備えて，インストール中にユーザの応答待ち状態になってから一定の時間が経過したあと，インストールを強制的に中断する時間（AIT ファイルの監視時間）を設定しておくことができます。

通常は，配布するソフトウェアをインストールするのに要した時間の約 3 倍の時間を目安に設定してください。

図 1-4 ［AIT ファイルの設定］パネル

パッケージングの詳細については，マニュアル「運用ガイド 1」の「2.1　パッケージングの方法」を参照してください。

1.2.4　リモートインストールの実行

リモートインストールマネージャで，ソフトウェアを配布するジョブを作成して実行してください。リモートインストール操作については，マニュアル「運用ガイド 1」の「2.3　リモートインストールの実行」を参照してください。

注意事項

クライアントが 07-00 より前のバージョンまたは UNIX システムの場合，AIT ファイルを使ったリモートインストールジョブはエラーになります。

1.3 AIT ファイルを作成および使用する場合の注意事項

AIT ファイルを作成および使用する場合の注意事項を次に示します。

- Web ページや，Java および ActiveX で作成されたソフトウェアには対応していません。
- AIT ファイルに，レジストリの HKEY_CURRENT_USER を操作する API を定義している場合は，リモートインストール先のクライアントに Administrator 権限を持っているユーザがログオンしている必要があります。
  Administrator 権限を持たないユーザは，HKEY_CURRENT_USER へのアクセス権がないため，API で操作するレジストリが自動的に HKEY_USERS¥.DEFAULT に変更されます。これによって，インストールがエラーになることがあります。
  なお，操作するレジストリが変更されても，インストール中のソフトウェアの動作に問題がない場合は，インストールが続行されます。
- インストール完了後にクライアントを再起動するよう設定されたパッケージを，AIT ファイルを使用してリモートインストールする場合，クライアントを再起動しないでインストーラが完了するよう設定した AIT ファイルを使用してください。リモートインストールを正常に完了させるには，パッケージの設定からクライアントを再起動する必要があります。
- 次に示すカテゴリの API の引数に 64 ビット関連のデータ（レジストリ，フォルダ，またはファイル）を指定して，64 ビット版の Windows 8，Windows Server 2012，64 ビット版の Windows 7，64 ビット版の Windows Server 2008，64 ビット版の Windows Vista または Windows Server 2003 (x64) 上で動作させた場合，API の操作対象が 32 ビット関連のデータに変更される（リダイレクトされる）ことがあります。
  - レジストリ操作
  - ディレクトリ操作
  - ファイル操作
  - INI ファイル操作

2 AIT ファイルの作成

Automatic Installation Tool は，AIT ファイルを作成するための統合された環境を提供しています。この章では，Automatic Installation Tool を使って，AIT ファイルを作成する方法について説明します。

2.1 Automatic Installation Tool の概要

Automatic Installation Tool の機能概要と，起動と終了方法について説明します。

2.1.1 Automatic Installation Tool の機能

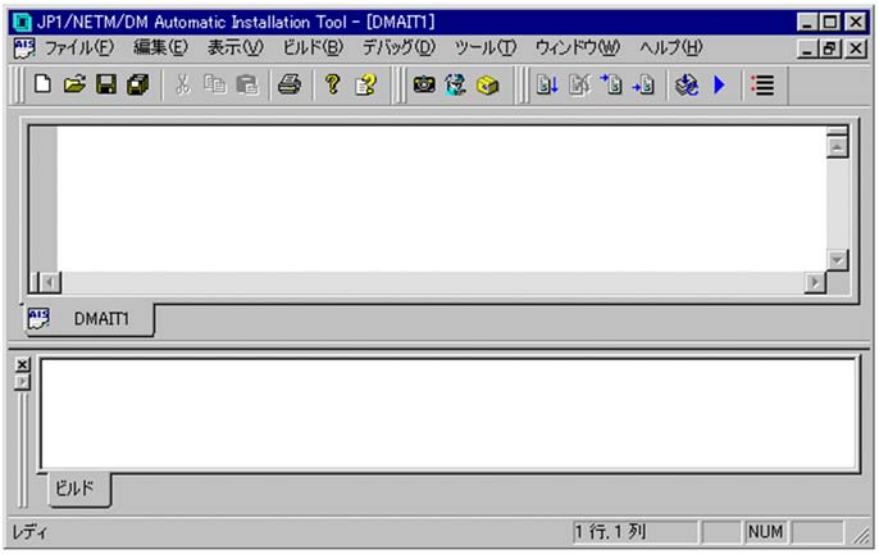
AIT ファイルを作成するために，Automatic Installation Tool は次の機能を備えています。

- 編集ウィンドウで，AIT ファイルを作成および編集します。テキストの切り取り，コピー，貼り付け，インデントの設定などができます。
- ウィンドウプロパティツールを使って，インストール画面のウィンドウ情報を取得できます。取得したウィンドウ情報はクリップボードにコピーし，AIT ファイル内で API の引数として使用できます。
- レコーダを使って，実際のインストール操作をレコーディングすると，ユーザ操作をシミュレートする AIT ファイルを自動生成できます。
- ［文法チェック］は，AIT ファイルが AIT 言語仕様に適合しているかどうかをチェックします。文法チェック時に発生したエラーは，アウトプットウィンドウに表示されます。
- ［実行］は，文法チェックをしたあと，AIT ファイルに記述されているインストール操作を再生します。
- ［デバッグ］は，文法の誤りの発見や修正を支援する機能を提供しています。AIT ファイル内に設定したブレークポイントまで実行したり，ステートメント単位に実行したりできます。また，AIT ファイル実行中に，変数の値を参照したり更新したりできます。
- インストール操作の再生，レジストリ処理，ファイル処理，文字列処理などのさまざまな処理を実行できるように，API が提供されています。
- 配布するソフトウェアを AIT ファイルに関連づけるための PP 識別情報ファイルを生成できます。

2.1.2 Automatic Installation Tool の起動と終了

Automatic Installation Tool を起動するには，JP1/NETM/DM のグループアイコンから［Automatic Installation Tool］を選択します。次のような［Automatic Installation Tool］ウィンドウが表示されます。

図 2-1 ［Automatic Installation Tool］ウィンドウ



Automatic Installation Tool を終了するには，［ファイル］－［終了］を選択します。

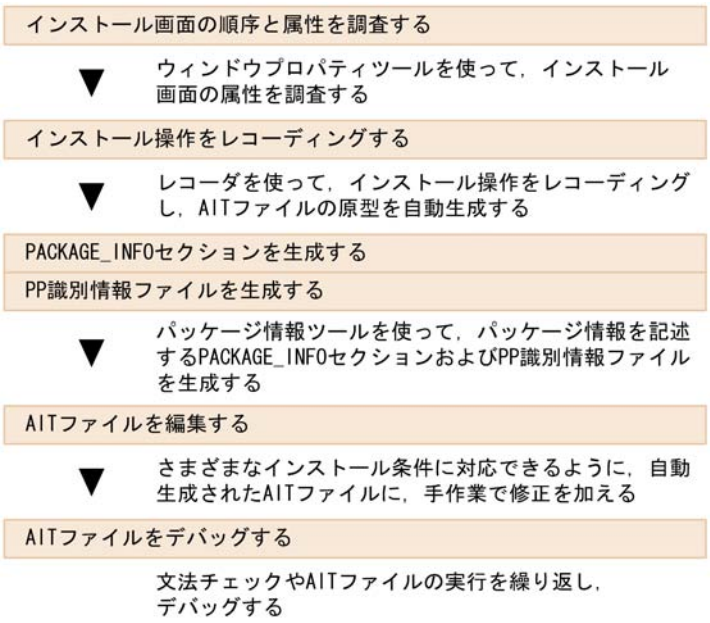
なお，Windows 8・Windows Server 2012・Windows 7・Windows Server 2008・Windows Vista 版 JP1/NETM/DM Client でレコーディングを実行する場合は，プログラムを実行する権限と同じ権限で［Automatic Installation Tool］ウィンドウを起動してください。

また，Windows 8・Windows Server 2012・Windows 7・Windows Server 2008・Windows Vista 版 JP1/NETM/DM Client は複数の［Automatic Installation Tool］ウィンドウを起動できます。ただし，レコーディングおよびデバッグを実行する場合は，一つのウィンドウから実行してください。

2.2 AIT ファイルの構造と作成手順

AIT ファイルは，基本的に次の手順で作成します。実際には，この手順を一度だけ実行するのではなく，何度か繰り返しながら AIT ファイルを完成させることになります。

図 2-2 AIT ファイルの作成手順



レコーダを使用してインストール操作をレコーディングしたあと，パッケージ情報ツールで PACKAGE_INFO セクションを生成すると，次の図のような構造の AIT ファイルが自動生成されます。自動生成されたファイルは，作成したい AIT ファイルの原型になります。通常は，網掛けの部分を手作業で修正し，AIT ファイルを完成させます。

図 2-3 AIT ファイルの構造

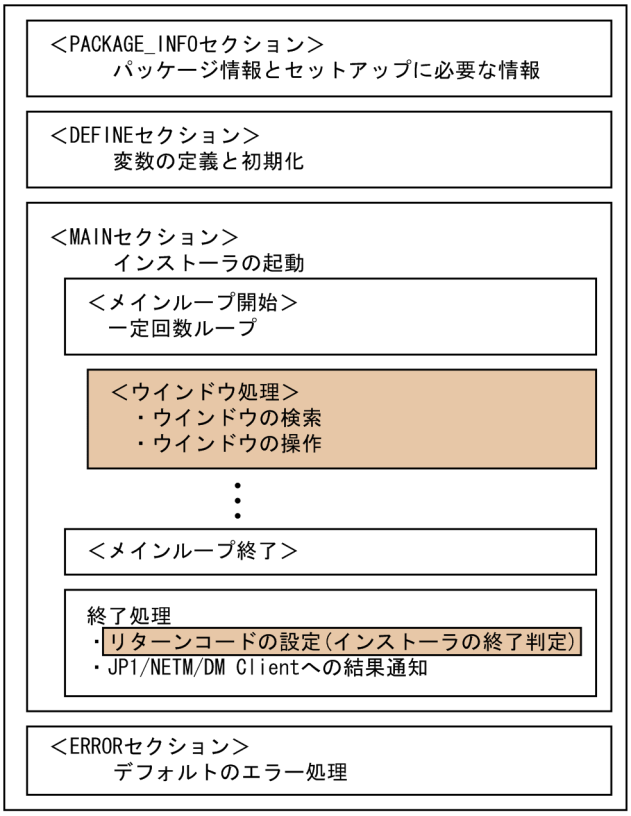


図 2-3 に示すように，AIT ファイルは四つのセクションで構成されています。すべてのセクションは必須で，セクションの順序は変更できません。次に，各セクションの概要を説明します。

● PACKAGE_INFO **セクション**

配布するソフトウェアの，パッケージ情報とセットアップに必要な情報を指定します。手作業でも作成できますが，パッケージ情報ツールを使って作成すると便利です。

● DEFINE **セクション**

MAIN セクションおよび ERROR セクションで使用する変数の定義と初期化を行います。ほかのセクションで変数を定義することはできません。MAIN セクションおよび ERROR セクションで使用する変数を追加したり，初期値を変更したりする場合は，このセクションを修正してください。

● MAIN **セクション**

インストーラが出力するウィンドウに対する操作を記述します。レコーダで自動生成されたコードを手作業で修正し，インストーラが出力するすべてのウィンドウに対する操作を記述します。インストール結果のリターンコードを設定することもできます。

● ERROR **セクション**

AIT ファイルの実行時に内部エラーが発生すると，このセクションに実行制御が移ります。エラーが発生したときの処理を変更する場合は，このセクションを修正してください。

AIT ファイル中には，コメントを記述できます。また，AIT 言語は大文字と小文字を区別しません。AIT 言語の詳細については，「3. AIT 言語リファレンス」を参照してください。

なお，PP 識別情報ファイルは手作業でも作成できますが，パッケージ情報ツールで生成すると便利です。PP 識別情報ファイルは，PACKAGE_INFO セクションを生成するときに合わせて生成できます。

パッケージ情報ツールによって生成された PP 識別情報ファイルは，JP1/NETM/DM のインストール先フォルダ ¥DMPRM に，PPDEFAIT.DMP というファイル名で格納されます。

2.3 インストール画面の順序と属性を調査する

配布するソフトウェアのインストーラを起動して，そのインストール手順を調査します。次の項目について，OS ごとにインストール方法を調べ，インストール手順を紙に記録してください。

- インストール画面の順序と属性
- 各ダイアログボックスの属性

これらの属性を調査するとき，Automatic Installation Tool の**ウィンドウプロパティ**ツールを利用すると便利です。ウィンドウプロパティツールは，ウィンドウおよびコントロールの GUI 属性を取得できるツールで，次の属性を取得できます。

ウィンドウ属性	説明
ウィンドウテキスト	ウィンドウまたはコントロールのキャプションです。
クラス名	ウィンドウまたはコントロールのクラス名です。
モジュール名	ウィンドウを起動したアプリケーションです。
コントロール ID	ウィンドウまたはコントロールの ID です。
コントロールの種類	ウィンドウ，ボタンなどのコントロールのタイプです。
関連付けるラベル	あるコントロールに対して，タブオーダーで一つ前のテキストが，そのコントロールの「関連付けるラベル」です。コントロールの一つ前にテキストがない場合，関連付けるラベルはありません。
使用可能	ウィンドウまたはコントロールが使用可能かどうかを識別します。
可視	ウィンドウまたはコントロールが見えるかどうかを識別します。

これらの属性は，ウィンドウやコントロールを識別するために，AIT ファイルの各種 API の入力値として使えます。

2.3.1 調査内容

インストーラが，ユーザに対して「どのような順序で，どういった操作を要求してくるか」を調査する必要があります。何度か手動でインストールして調査し，次の項目について一覧表にまとめてください。

インストール方法や，インストールする PC の状態（OS の種類，ハードディスクの空き容量，メモリの空き容量，インストール済みのソフトなど）によって，インストールの操作が変わりますので，入念に調査してください。

なお，配布するソフトウェアのインストールで再起動が発生する場合，再起動後の操作については，AIT ファイルではサポートしていません。したがって，再起動開始時点までのインストール操作を調査してください。

● ウィンドウテキスト
ウィンドウプロパティツールを使って調査した「ウィンドウテキスト」を記述します。AIT 言語は全角と半角を区別します。大文字と小文字は区別しません。

● クラス名
ウィンドウプロパティツールを使って調査した「クラス名」を記述します。

● コントロール ID
ウィンドウプロパティツールを使って調査した「コントロール ID」を記述します。

● コントロールの種類

ウィンドウプロパティツールを使って調査した「コントロールの種類」を記述します。

● 操作

このダイアログボックスに対する操作（［OK］ボタンをクリックするなど）を記述します。

● 備考

コメントや特記事項がある場合に記述します。

次に，例として，Acrobat Reader 5.05 のインストールの流れを調査した結果を示します。「コントロール ID」欄の N/A は，コントロール ID が適用されていないことを示しています。

表 2-1　Acrobat Reader 5.05 のインストールの流れ

#	ウィンドウテキスト	クラス名	コントロール ID	コントロールの種類	操作	備考
1	Acrobat Reader 5.0.5 のセットアップ	#32770	N/A	Window	• ［次へ］ボタンをクリックする。	なし
2	インストール先の選択	#32770	N/A	Window	• 1 回目は［参照］ボタンをクリックする。 • 2 回目は［次へ］ボタンをクリックする。	2 回目は #4 のあとへ
3	ディレクトリの選択	#32770	N/A	Window	• パス名にインストールパスを指定する。	インストール先を変更する場合だけ表示
4	セットアップ	#32770	N/A	Window	• ［OK］ボタンをクリックする。	インストール先を変更する場合だけ表示
5	セットアップの完了	#32770	N/A	Window	• ［いいえ，後でコンピュータを再起動します］を選択する。 • ［完了］ボタンをクリックする。	なし

2.3.2　インストール画面の属性の取得

ウィンドウプロパティツールを使って，インストール画面の属性を取得する方法について説明します。

(1) ウィンドウおよびコントロールの属性を取得する

1. **［ツール］－［ウィンドウプロパティ］を選択する。**
   ［ウィンドウプロパティ］ダイアログボックスが表示されます。
2. **属性を取得したいソフトウェアを起動する。**
   調べたいウィンドウと［ウィンドウプロパティ］ダイアログボックスが，ともにデスクトップに表示されていることを確認してください。
3. **調べたいウィンドウまたはコントロールの上に，ウィンドウプロパティツールのファインダ（  ）をドラッグ＆ドロップする。**
   ［ウィンドウプロパティ］ダイアログボックスに，ファインダをドラッグした先のウィンドウまたはコントロールの属性が表示されます。

図 2-4 ［ウィンドウプロパティ］ダイアログボックス

ウィンドウプロパティツールで取得した属性は，クリップボードにコピーできます。

［ウィンドウプロパティ］ダイアログボックスの をクリックしてください。属性は次のようにコピーされます。

```
ウィンドウテキスト：Acrobat Reader 5.0.5 のセットアップ
クラス名：#32770
モジュール名：_INS5576._MP
コントロール ID：N/A
コントロールの種類：Window
関連付けるラベル：N/A
使用可能：はい
可視：はい
```

(2) 関連するウィンドウおよびコントロールの属性を表示する

［ウィンドウプロパティ］ダイアログボックス上のツールバーのボタンを使って，親ウィンドウや最初の子ウィンドウの属性を表示したり，前面や背面のウィンドウの属性を表示したりできます。また，［ウィンドウプロパティ］ダイアログボックスの表示を調整できます。次に，［ウィンドウプロパティ］ダイアログボックス上のツールバーのボタンを示します。

ツールバーのボタン	説明
	親ウィンドウにジャンプ：属性を取得したウィンドウまたはコントロールの，親ウィンドウの属性を表示します。
	最初の子ウィンドウにジャンプ：属性を取得したウィンドウまたはコントロールの，最初の子ウィンドウの属性を表示します。
	前のウィンドウにジャンプ：属性を取得したウィンドウまたはコントロールの，一つ前面のウィンドウの属性を表示します。
	次のウィンドウにジャンプ：属性を取得したウィンドウまたはコントロールの，一つ背面のウィンドウの属性を表示します。
	検索中に表示／非表示：ファインダをドラッグしたとき，［ウィンドウプロパティ］ダイアログボックスを非表示にし，ドロップしたときに［ウィンドウプロパティ］ダイアログボックスを表示することができます。非表示にしておくと，対象のコントロールを選択しやすくなります。
	常にトップに：［ウィンドウプロパティ］ダイアログボックスを，常に最前面に表示します。

2.4 インストール操作をレコーディングする

Automatic Installation Tool の**レコーダ**を使用して，実際のインストール操作をレコーディングします。レコーダは，キーを押す，マウスをクリックする，コントロールの操作などのイベントを記録し，ユーザ操作をシミュレートする AIT ファイルを自動的に作成します。この自動生成されたファイルは，作成したい AIT ファイルの原型になります。

レコーダによって，PACKAGE_INFO セクションを除く，すべてのセクションが自動生成されます。Acrobat Reader 5.05 のインストール操作をレコーディングした結果を例に，どのような情報が生成されるのかを次の図に示します。

図 2-5 Acrobat Reader 5.05 のレコーディングで自動生成された AIT ファイル（1/3）

```
DEFINE
{
    integer iLoopCount = 0;
    integer iLoopMax = 60;
    integer DM_RTN;
    integer WINH;
    integer iCapsLockState;
    integer iNumLockState;
    integer iScrollLockState;
    integer iInsertLockState;
    integer AITIGNORE = 0;
    integer AITFLAG1=1;
    integer AITFLAG2=1;
    integer AITEVENTFLAG1=1;
    bool bRtn;
    const integer OK_END = 0;
    const integer NG_END = -1;
    float SLEEP_TIME = 1.0;
    float SLEEP_TIME_RESTART = 10.0;
    float SLEEP_TIME_EVENTS = 0.5;
}
MAIN
{
    AIT_SetDefaultWaitTimeout(1.0);
    AIT_DMPSTRC();
    DM_RTN = NG_END;
    iCapsLockState = AIT_GetKeyState(CAPSLOCK);
    iNumLockState = AIT_GetKeyState(NUMLOCK);
    iScrollLockState = AIT_GetKeyState(SCROLLLOCK);
    iInsertLockState = AIT_GetKeyState(INSERTLOCK);
    bRtn= AIT_Exec(InstallerName,SW_SHOWNORMAL);
    if(bRtn == false)
        iLoopCount = iLoopMax;
    Endif;
```

インストーラの起動

図 2-6　Acrobat Reader 5.05 のレコーディングで自動生成された AIT ファイル（2/3）

```
while(iLoopCount < iLoopMax)
    if(AIT_FocusWindow("Acrobat Readerを解凍中です...", "#32770") > 0)
        AIT_Sleep(SLEEP_TIME);
        iLoopCount=0;
    Endif;
    if(AIT_FocusWindow("セットアップ", "#32770") > 0)
        if((AITFLAG1==1)&&(AITIGNORE == 0))
            AIT_Sleep(SLEEP_TIME);
            AITFLAG1=2;
            AITIGNORE = 1;
        Endif;
        if((AITFLAG1==2)&&(AITIGNORE == 0))
            AIT_Sleep(SLEEP_TIME);
            AITFLAG1=0;
            AITIGNORE = 1;
        Endif;
        AITIGNORE=0;
        iLoopCount=0;
    Endif;
    if(AIT_FocusWindow("Acrobat Reader のセットアップ", "InstallShield_Win") > 0)
        if((AITFLAG2==1)&&(AITIGNORE == 0))
            AIT_Sleep(SLEEP_TIME);
            AITFLAG2=2;
            AITIGNORE = 1;
        Endif;
        if((AITFLAG2==2)&&(AITIGNORE == 0))
            AIT_Sleep(SLEEP_TIME);
            AITFLAG2=3;
            AITIGNORE = 1;
        Endif;
        AITIGNORE=0;
        iLoopCount=0;
    Endif;
    if(AIT_FocusWindow("Acrobat Reader 5.0.5 のセットアップ", "#32770") > 0)
        AIT_CtrlSetFocus("次へ(&N) >",BUTTON_CTRL);
        AIT_Sleep(SLEEP_TIME_EVENTS);
        AIT_SetKeyState(CAPSLOCK, KEYSTATE_OFF);
        AIT_Sleep(SLEEP_TIME_EVENTS);
        AIT_SetKeyState(NUMLOCK, KEYSTATE_ON);
        AIT_Sleep(SLEEP_TIME_EVENTS);
        AIT_SetKeyState(SCROLLLOCK, KEYSTATE_OFF);
        AIT_Sleep(SLEEP_TIME_EVENTS);
        AIT_SetKeyState(INSERTLOCK, KEYSTATE_OFF);
        AIT_Sleep(SLEEP_TIME_EVENTS);
        AIT_PlayKey("{ALT_LOCK}n{ALT_UNLOCK}");
        AIT_Sleep(SLEEP_TIME_EVENTS);
        iLoopCount=0;
    Endif;
    if(AIT_FocusWindow("インストール先の選択", "#32770") > 0)
        AIT_CtrlSetFocus("次へ(&N) >",BUTTON_CTRL);
        AIT_Sleep(SLEEP_TIME_EVENTS);
        AIT_PlayKey("{ALT_LOCK}n{ALT_UNLOCK}");
        AIT_Sleep(SLEEP_TIME_EVENTS);
        iLoopCount=0;
    Endif;
```

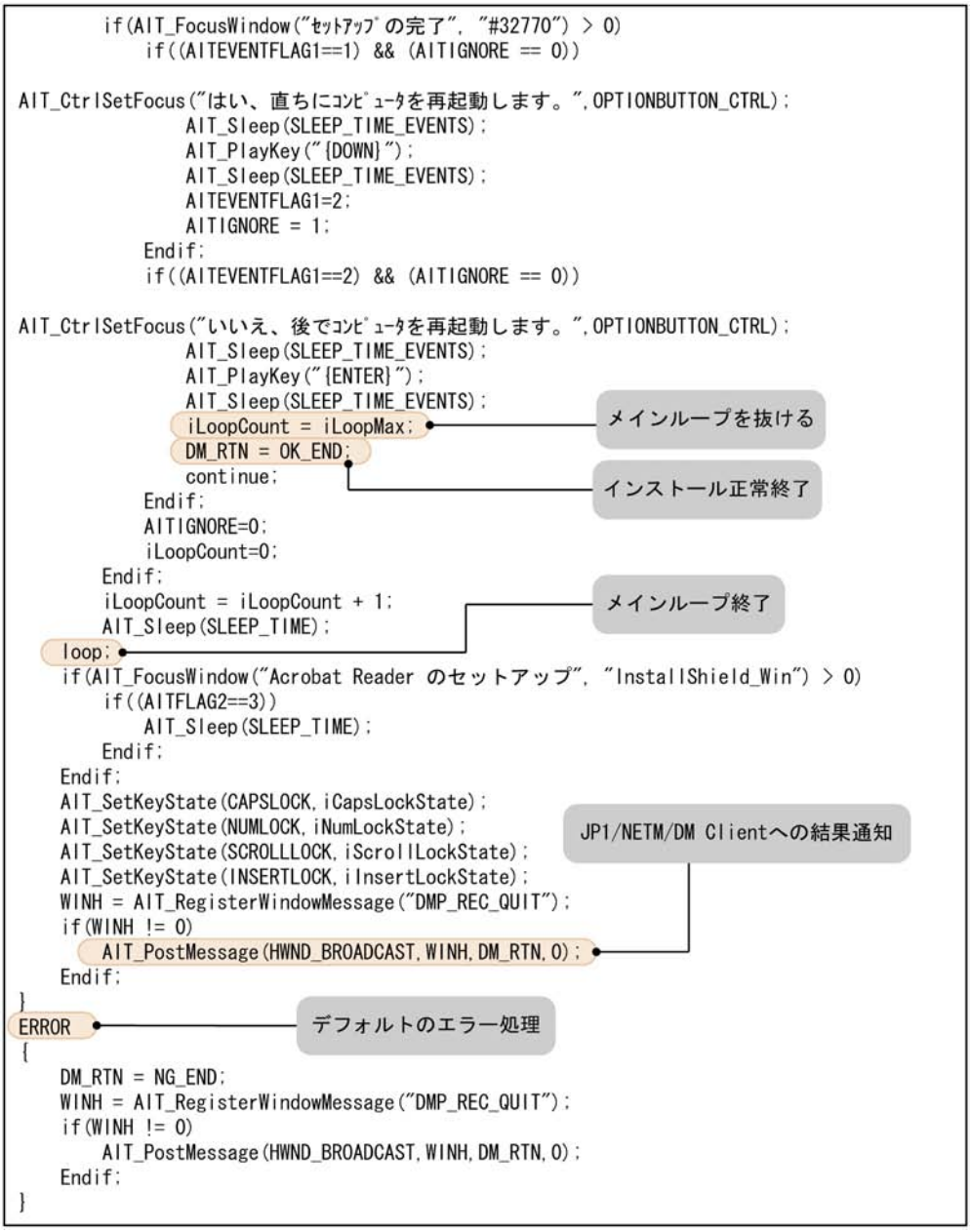
メインループ開始
一定回数ループ

ウィンドウ処理
（ウィンドウの検索と操作）

ウィンドウ情報
キャプション：Acrobat Reader5.0.5のセットアップ
クラス　　：#32770

[Alt] ＋ [n] キーを押す

図 2-7　Acrobat Reader 5.05 のレコーディングで自動生成された AIT ファイル（3/3）




```
        if(AIT_FocusWindow("ｾｯﾄｱｯﾌﾟの完了", "#32770") > 0)
            if((AITEVENTFLAG1==1) && (AITIGNORE == 0))

AIT_CtrlSetFocus("はい、直ちにｺﾝﾋﾟｭｰﾀを再起動します。",OPTIONBUTTON_CTRL);
                AIT_Sleep(SLEEP_TIME_EVENTS);
                AIT_PlayKey("{DOWN}");
                AIT_Sleep(SLEEP_TIME_EVENTS);
                AITEVENTFLAG1=2;
                AITIGNORE = 1;
            Endif;
            if((AITEVENTFLAG1==2) && (AITIGNORE == 0))

AIT_CtrlSetFocus("いいえ、後でｺﾝﾋﾟｭｰﾀを再起動します。",OPTIONBUTTON_CTRL);
                AIT_Sleep(SLEEP_TIME_EVENTS);
                AIT_PlayKey("{ENTER}");
                AIT_Sleep(SLEEP_TIME_EVENTS);
                iLoopCount = iLoopMax;
                DM_RTN = OK_END;
                continue;
            Endif;
            AITIGNORE=0;
            iLoopCount=0;
        Endif;
        iLoopCount = iLoopCount + 1;
        AIT_Sleep(SLEEP_TIME);
    loop;
    if(AIT_FocusWindow("Acrobat Reader のセットアップ", "InstallShield_Win") > 0)
        if((AITFLAG2==3))
            AIT_Sleep(SLEEP_TIME);
        Endif;
    Endif;
    AIT_SetKeyState(CAPSLOCK,iCapsLockState);
    AIT_SetKeyState(NUMLOCK,iNumLockState);
    AIT_SetKeyState(SCROLLLOCK,iScrollLockState);
    AIT_SetKeyState(INSERTLOCK,iInsertLockState);
    WINH = AIT_RegisterWindowMessage("DMP_REC_QUIT");
    if(WINH != 0)
        AIT_PostMessage(HWND_BROADCAST,WINH,DM_RTN,0);
    Endif;
}
ERROR
{
    DM_RTN = NG_END;
    WINH = AIT_RegisterWindowMessage("DMP_REC_QUIT");
    if(WINH != 0)
        AIT_PostMessage(HWND_BROADCAST,WINH,DM_RTN,0);
    Endif;
}
```

2.4.1　インストール操作のレコーディング

レコーダを使用して，実際のインストール操作をレコーディングする方法について説明します。レコーディングによって，ユーザ操作をシミュレートする AIT ファイルの原型が自動生成されます。

レコーダでインストール操作を記録するとき，AIT ファイルに自動ロギング機能（AIT_LogMesssage ステートメント）を追加することもできます。これによって，記録されたイベントを再生するとき，ログメッセージが出力されます。レコーダを使用して AIT ファイルを作成したあと，［ビルド］－［実行］で，この AIT ファイルを実行し，テスト対象のアプリケーションが，シミュレートされたユーザ操作と同じように動作するかどうかを検証するときに役立ちます。

なお，インストール操作では，マウスをできるだけ使用しないでください。マウスからの操作は，画面座

標に依存してしまうため，AIT ファイルからインストーラへの確実な応答ができなくなってしまいます。また，マウスのホイールを利用した操作は，正しくレコーディングされない場合があります。そのため，レコーディングするときは，マウス操作ではなく，キーボード入力のイベントを記録するようにしてください。

レコーディングするときは，Automatic Installation Tool とインストーラ以外のすべてのアプリケーションを終了させておくことをお勧めします。別のアプリケーションに対する操作を行わない場合でも，レコーダは表示されているすべてのウィンドウを記録します。

1. **［ツール］メニューから［レコーダ］を選択する。**
   ［レコーダ］ダイアログボックスが表示されます。

図 2-8 ［レコーダ］ダイアログボックス（レコーディング開始）

2. **「インストールプログラム名」および「ログに出力する」を指定する。**

   インストールプログラム名

   インストール操作を記録する前にレコーダによって呼び出される実行プログラムの名前を指定します。インストールプログラム名を指定しない場合は，あらかじめ，対象のアプリケーションを起動しておいてください。

   ログに出力する

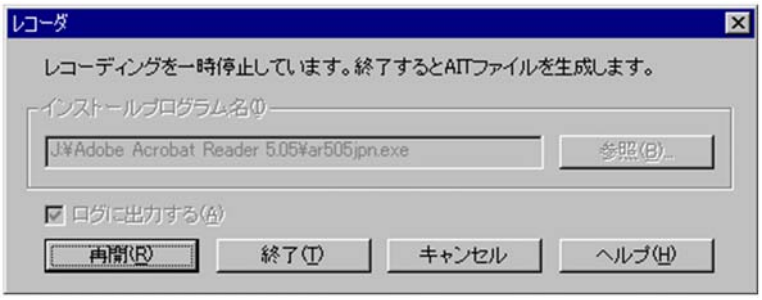
   オンにすると，AIT ファイルに AIT_LogMesssage ステートメントが追加され，AIT ファイルを再生したときに，エラーや情報メッセージが記録されます。
   オフにすると，AIT ファイルは作成されますが，メッセージを記録するためのステートメントは追加されません。

3. **［開始］ボタンをクリックする。**
   これ以降のユーザ操作が記録されます。
   操作 2 で「インストールプログラム名」を指定していた場合，指定したインストールプログラムが呼び出されます。

4. **実際に，ソフトウェアのインストール操作を行う。**
   ユーザ操作をシミュレートする記録シーケンスが作成されます。
   ユーザ操作が記録されている間，［Automatic Installation Tool］のアイコンが Windows のタスクバーに表示されます。

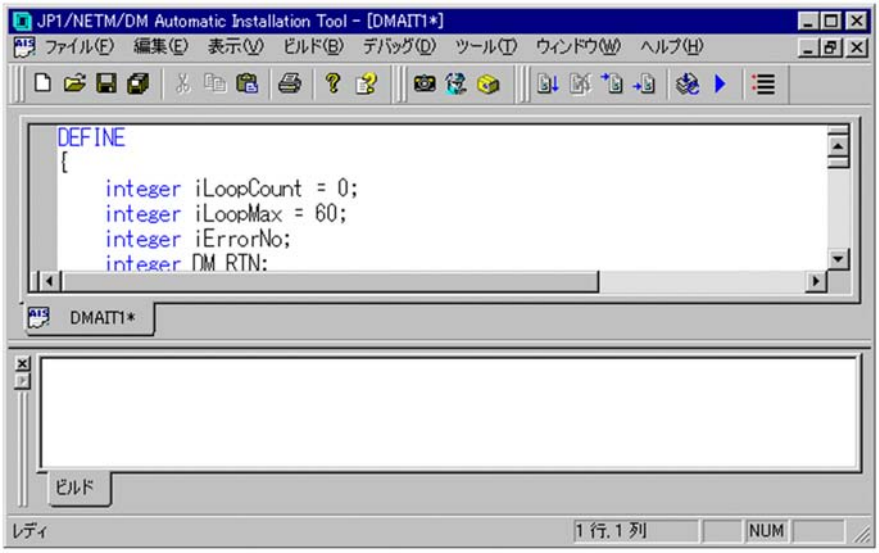
5. **ソフトウェアのインストール作業が終わったら，Windows のタスクバーで［Automatic Installation Tool］のアイコンをクリックする。**
   レコーディングの終了は基本的に左クリックで行ってください。
   Windows 8・Windows Server 2012・Windows 7・Windows Server 2008 R2 上で右クリックした場合，Windows のジャンプリストが表示されインストール操作としてレコーディングされるため実行しないでください。
   ［レコーダ］ダイアログボックスが表示され，レコーディングは一時停止の状態になります。

図 2-9 ［レコーダ］ダイアログボックス（レコーディング一時停止）



6. **［終了］ボタンをクリックする。**
   レコーディングが終了し，パッケージ情報を更新するかどうかを確認するメッセージが表示されます。［はい］をクリックすると，［パッケージ情報］ダイアログボックスが表示され，引き続き PACKAGE_INFO セクションを生成することができます。［パッケージ情報］ダイアログボックスについては，「2.5　PACKAGE_INFO セクションを生成する」を参照してください。
   ［いいえ］をクリックすると，［Automatic Installation Tool］ウィンドウ中に，作成された AIT ファイルが表示されます。

図 2-10　AIT ファイルが表示された［Automatic Installation Tool］ウィンドウ



7. **［ファイル］－［名前を付けて保存］を選択し，自動作成された AIT ファイルに名前を付けて保存する。**
   AIT ファイルの拡張子は .ais です。このファイルを基に，必要なコーディングを追加していきます。

なお，レコーディング中に再起動イベントが発生した場合，レコーディング中にユーザによって実行された操作は Automatic Installation Tool で保持されています。PC を再起動して Automatic Installation Tool を起動したあと，AIT ファイルの生成を確認するメッセージが表示されますので，［はい］をクリックして AIT ファイルを生成してください。

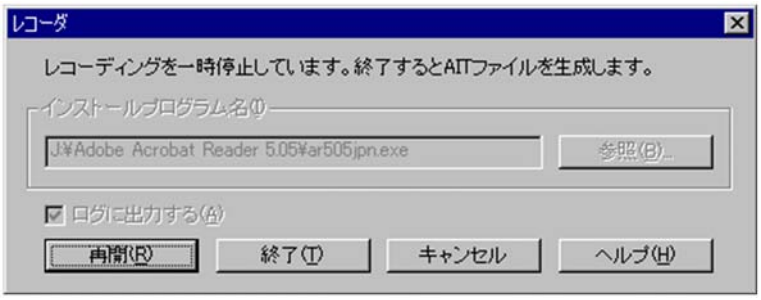
2.4.2　レコーディングの一時停止と再開

インストール操作のレコーディング中に，記録を一時停止したり，再開したりできます。操作方法を次に示します。

1. **Windows のタスクバーで［Automatic Installation Tool］のアイコンを選択する。**

［レコーダ］ダイアログボックスが表示され，レコーディングは一時停止の状態になります。

図 2-11 ［レコーダ］ダイアログボックス（レコーディング一時停止）



2. **レコーディングが一時停止している状態で，［再開］ボタンをクリックする。**
   レコーディングが再開されます。インストール操作を続行してください。

2.4.3 再起動を伴うインストール操作のレコーディング

OS の再起動を要求されるインストール操作のレコーディングでは，OS が再起動するタイミングでレコーディングが完了します。

生成された AIT ファイルを表示するには，OS の再起動後に Automatic Installation Tool を起動してください。Automatic Installation Tool を起動すると，再起動前にレコーディングした内容を有効にするかどうかを選択するメッセージダイアログボックスが表示されます。

図 2-12 レコーディング内容の有効 / 無効を選択するメッセージダイアログボックス

［はい］ボタンをクリックすると，再起動前にレコーディングした内容の AIT ファイルが表示されます。

2.5 PACKAGE_INFO セクションを生成する

AIT ファイルには，配布するソフトウェアのパッケージ情報とセットアップに必要な情報を指定する，PACKAGE_INFO セクションが必要です。この PACKAGE_INFO セクションを作成，検証するには，**パッケージ情報**ツールを使うと便利です。

また，AIT ファイルを利用するためには，配布するソフトウェアを AIT ファイルに関連づけるための PP 識別情報ファイルが必要です。PP 識別情報ファイルは，パッケージ情報ツールから自動で生成できます。生成した PP 識別情報ファイルを編集する方法については，「付録 C　PP 識別情報ファイルの編集方法」を参照してください。

ここでは，パッケージ情報ツールを使用して PACKAGE_INFO セクションおよび PP 識別情報ファイルを生成する方法について説明します。

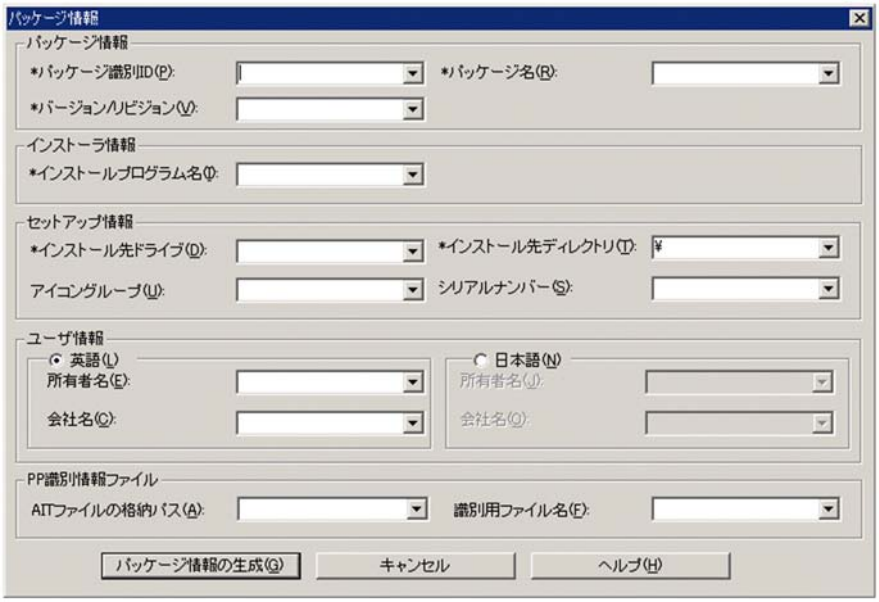
2.5.1 PACKAGE_INFO セクションおよび PP 識別情報ファイルの生成手順

PACKAGE_INFO セクションおよび PP 識別情報ファイルを生成する手順について説明します。

PACKAGE_INFO セクションの詳細については，「3.2.1　PACKAGE_INFO」を参照してください。

1. **［ファイル］－［開く］を選択し，対象の AIT ファイルを開く。**
2. **［ツール］－［パッケージ情報］を選択する。**
   ［パッケージ情報］ダイアログボックスが表示されます。

図 2-13 ［パッケージ情報］ダイアログボックス



AIT ファイルに PACKAGE_INFO セクションがすでにある場合は，該当する値が［パッケージ情報］ダイアログボックスに表示されます。

3. **各項目に値を入力する。**
   どの項目にも，¥n，¥r，¥t のような特殊な意味を持つ文字は使用できません。また，ダイアログボックス中の各項目の先頭の * は，必ず指定する項目であることを示しています。各項目の意味は次のと

おりです。

項目	説明
パッケージ識別 ID	パッケージ識別 ID を，1 ～ 44 バイトで指定します。 半角英数字（英字は大文字だけ），「-」（ハイフン），および「_」（アンダーバー）が使用できます。
パッケージ名	パッケージ名を，1 ～ 50 バイトで指定します。「¥」と「;」は使用できません。
バージョン／リビジョン	ソフトウェアのバージョン / リビジョンを，1 ～ 6 バイトで指定します。 英数字（英字は大文字だけ），および「/」（スラッシュ）が使用できます。
インストールプログラム名	ソフトウェアをインストールするときの，インストールプログラム（インストーラ）の名前を，1 ～ 256 バイトで指定します。 次の記号は使用できません。 「*」「"」「:」「|」「<」「>」「?」
インストール先ドライブ	ソフトウェアをインストールするドライブを，2 バイトで指定します。 半角英数字および「:」（コロン）で指定します。
インストール先ディレクトリ	ソフトウェアをインストールするディレクトリを指定します。¥ で始まるパス名を，1 ～ 128 バイトで指定します。
アイコングループ	ソフトウェアのアイコングループを，1 ～ 40 バイトで指定します。
シリアルナンバー	インストールするソフトウェアのシリアルナンバーを，1 ～ 64 バイトで指定します。インストール時に CD キーの必要なソフトウェアは，CD キーを入力してください。
英語	ソフトウェアが英語版の場合はオンにし，以下の「所有者名」と「会社名」を指定します。
所有者名	ソフトウェアの所有者名を，1 ～ 40 バイトで指定します。 半角英数字で指定します。
会社名	ソフトウェアを所有する会社名を，1 ～ 80 バイトで指定します。 半角英数字で指定します。
日本語	ソフトウェアが日本語版の場合はオンにし，以下の「所有者名」と「会社名」を指定します。
所有者名	ソフトウェアの所有者名を，1 ～ 40 バイトで指定します。
会社名	ソフトウェアを所有する会社名を，1 ～ 80 バイトで指定します。
AIT ファイルの格納パス	生成する AIT ファイルのフルパスを，ドライブ名も含めて 1 ～ 256 バイトで指定します。 「;」は使用できません。
識別用ファイル名	配布するソフトウェアをユニークに識別できるファイル名を，1 ～ 477 バイトで指定します。 パッケージング時に指定したファイルが存在すると，AIT ファイルを利用して配布するソフトウェアであると判断されます。複数のファイルを指定した場合は，すべてのファイルが存在するときだけ配布するソフトウェアであると判断されます。複数のファイルを指定する場合，ファイル名を「;」で区切って指定します。

なお，「AIT ファイルの格納パス」と「識別用ファイル名」の両方を指定しないで PACKAGE_INFO セクションを生成した場合は，PP 識別情報ファイルは生成されません。

また，「インストールプログラム名」と「識別用ファイル名」の指定方法については，「2.5.2　インストールプログラムと識別用ファイルの指定方法」を参照してください。

4. **［パッケージ情報の生成］ボタンをクリックする。**

各項目の長さ，無効な文字，必須項目など，パッケージャと同様の検証が行われ，AIT ファイル内に PACKAGE_INFO セクションが生成，更新されます。

```
PACKAGE_INFO
{
    PackageID         = "ADOBEACROBATREADER";
    Product           = "Adobe Acrobat Reader";
    Version           = "505";
    InstallerName     = "ar505jpn.exe";
    InstallDrive      = "C:";
    InstallDirectory  = "'¥Program Files'¥Adobe";
}
```

また，［パッケージ情報］ダイアログボックスに設定された値で，PP 識別情報ファイルが生成されます。すでに PP 識別情報ファイルが存在する場合は，PP 識別情報ファイルに情報が追加されます。PP 識別情報ファイルは，JP1/NETM/DM のインストール先フォルダ ¥DMPRM 下に，PPDEFAIT.DMP というファイル名で格納されます。

5. **［ファイル］－［上書き保存］を選択し，生成，更新された** PACKAGE_INFO **セクションを，**AIT **ファイルに保存する。**

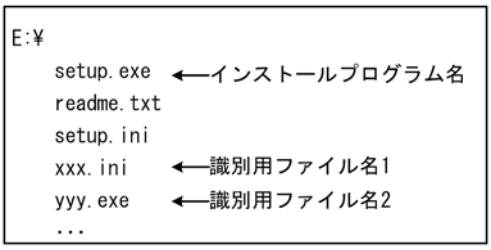
［パッケージ情報］ダイアログボックスで入力した値は，パッケージング時に［JP1/NETM/DM パッケージング］ダイアログボックスに表示されます。このとき，「パッケージ識別 ID」，「パッケージ名」，および「バージョン／リビジョン」の値は変更できません。そのほかの値は，パッケージング時またはリモートインストール時に変更することもできます。

2.5.2 インストールプログラムと識別用ファイルの指定方法

AIT ファイルで指定する「インストールプログラム名」と PP 識別情報ファイルで指定する「識別用ファイル名」は，パッケージングするディレクトリからの相対パスで指定する必要があります。

次に，パッケージングするファイルが CD-ROM ドライブ（E:）に入っていると仮定して，AIT ファイルおよび PP 識別情報ファイルでの指定方法を説明します。

● パッケージングするディレクトリにサブディレクトリが存在しない場合



パッケージングするディレクトリ

```
E:¥
```

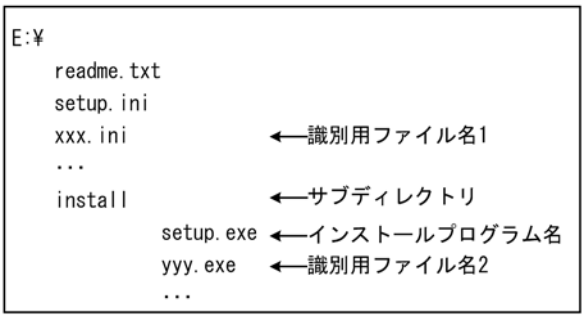
AIT ファイルのパッケージ情報で指定する「インストールプログラム名」

```
setup.exe
```

PP 識別情報ファイルで指定する「識別用ファイル名」

```
xxx.ini;yyy.exe
```

● パッケージングするディレクトリにサブディレクトリが存在する場合



パッケージングするディレクトリ

```
E:¥
```

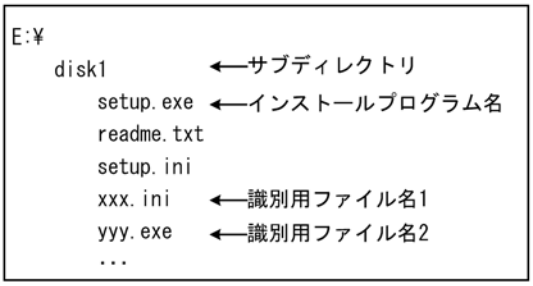
AIT ファイルのパッケージ情報で指定する「インストールプログラム名」

```
install¥setup.exe
```

PP 識別情報ファイルで指定する「識別用ファイル名」

```
xxx.ini;install¥yyy.exe
```

● パッケージングするディレクトリがサブディレクトリの場合



パッケージングするディレクトリ

```
E:¥disk1
```

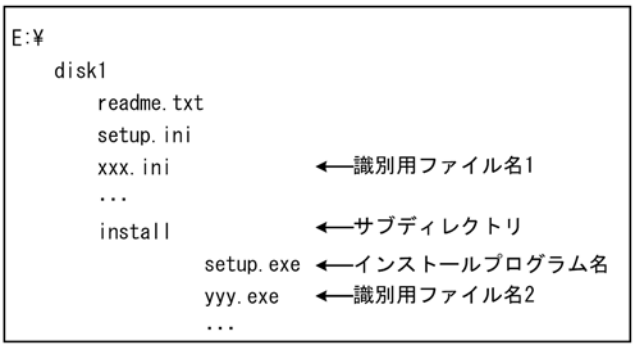
AIT ファイルのパッケージ情報で指定する「インストールプログラム名」

```
setup.exe
```

PP 識別情報ファイルで指定する「識別用ファイル名」

```
xxx.ini;yyy.exe
```

● パッケージングするディレクトリがサブディレクトリで，その下にサブディレクトリが存在する場合



パッケージングするディレクトリ

```
E:¥disk1
```

AIT ファイルのパッケージ情報で指定する「インストールプログラム名」

```
install¥setup.exe
```

PP 識別情報ファイルで指定する「識別用ファイル名」

```
xxx.ini;install¥yyy.exe
```

2.6 AIT ファイルを編集する

レコーディングで AIT ファイルの原型が自動生成されたあと，AIT ファイルを手作業で編集します。編集のためには，AIT ファイルで頻繁に使用されるウィンドウ処理の API について理解しておく必要があります。

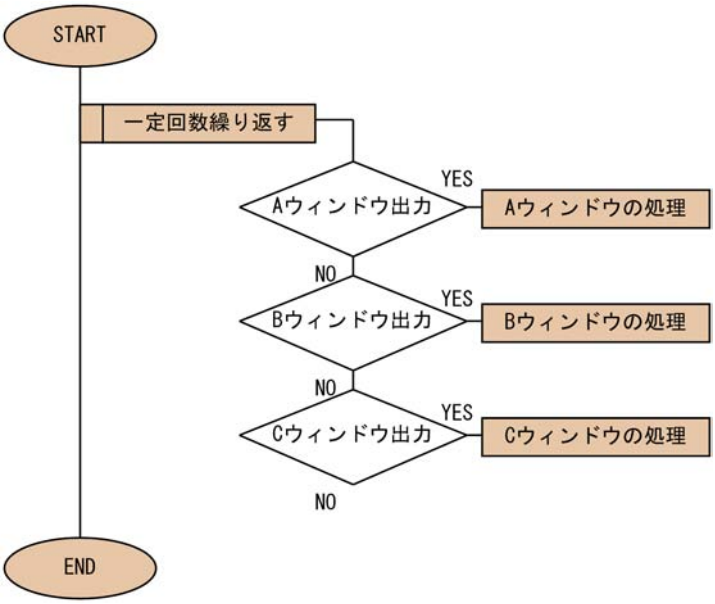
ここでは，ウィンドウ処理の API について説明したあと，手作業でどのような修正を加えたらよいのかを説明します。

2.6.1 ウィンドウ処理について

MAIN セクションでは，表示されたダイアログボックスに対して，クライアントユーザが誤って応答したり，キーボードやマウスを操作したりしたために，インストールが中断するような事態に備えて，メイン処理の部分を作成する必要があります。

AIT ファイルでのウィンドウ処理を次の図に示します。

図 2-14 AIT ファイルでのウィンドウ処理



AIT ファイルは，ループ中で繰り返しウィンドウを検索します。したがって，クライアントユーザが誤って A ウィンドウに応答して B ウィンドウが表示された状態でも，AIT ファイルは A ウィンドウの処理をスキップして B ウィンドウの処理を実行できます。

AIT ファイルを作成する場合は，出力されるウィンドウを調査し，そのウィンドウに対する処理を並べます。このような処理構造によって，出力されるウィンドウの順序やユーザの操作に関係なく処理を完結できます。

AIT ファイルでのウィンドウ処理は，次の処理を一つのパーツとしてループ中にシーケンシャルに並べます。

- ウィンドウの検索
- ウィンドウの操作

これらのウィンドウ処理では，表 2-2 および表 2-3 に示す API を使用します。

表 2-2 ウィンドウの検索で使用する API

API 名称	説明
AIT_FocusWindow	ウィンドウを検索し，フォーカスを設定します。
AIT_CtrlSetFocus	特定のコントロールにフォーカスを設定します。

表 2-3 ウィンドウの操作で使用する API

API 名称	説明
AIT_VerifyExistence	ウィンドウ中に，ボタンやチェックボックスなどのコントロールがあるかないかを確認します。
AIT_VerifyEnabled	コントロールが使用可能かどうかを確認します。
AIT_VerifyPos	コントロールの位置を確認します。
AIT_PlayKey	［Enter］キーを押すなどの，キーボード操作をシミュレートします。

ウィンドウ処理の注意事項

Windows 8，Windows Server 2012，Windows 7，Windows Server 2008，Windows Vista，Windows Server 2003，Windows XP，または Windows 2000 環境で，AIT ファイルを使用してソフトウェアをリモートインストールする場合，ウィンドウ操作の API（AIT_FocusWindow および AIT_CtrlSetFocus）ではアプリケーションにフォーカスが設定できないことがあります。その際は，レコーダ操作の API である AIT_Exec または AIT_ExecCommand の実行後に，AIT_PlayKey で［Alt］+［Tab］キーをシミュレートし，フォーカスをデスクトップから移動してください。

次に，これらの API の役割と使用方法について説明します。各 API のパラメタおよび戻り値の詳細については，「4. API リファレンス」を参照してください。

(1) AIT_FocusWindow

AIT_FocusWindow はウィンドウを検索し，そのウィンドウにフォーカスを設定します。
AIT_FocusWindow でウィンドウを検索してフォーカスを設定したあと，ウィンドウに対する処理を行います。

ウィンドウを検索するためには，パラメタとして，ウィンドウテキストとクラス名を指定します。ウィンドウテキストとクラス名を調査するには，ウィンドウプロパティツールを使用すると便利です。また，レコーダで自動生成した AIT ファイルのコードを利用する方法もあります。

戻り値は，ウィンドウの検索に成功した場合はウィンドウハンドル，検索に失敗した場合は 0 になります。

次に，AIT_FocusWindow の使用例を示します。

図 2-15 AIT_FocusWindow の使用例

```
if(AIT_FocusWindow("ﾃﾞｨﾚｸﾄﾘの選択", "#32770") > 0)
    AIT_PlayKey("%(p)");
    AIT_PlayKey(InstallPoint);
    AIT_PlayKey("{ENTER}");
    AIT_Sleep(SLEEP_TIME_EVENTS);
    iLoopCount = 0;
Endif;
```

(2) AIT_CtrlSetFocus

AIT_CtrlSetFocus は，特定のコントロールにフォーカスを設定します。ウィンドウの中に複数のコントロールがある場合は，AIT_CtrlSetFocus でコントロールにフォーカスを設定し，そのあとユーザ操作をシミュレートします。

コントロールを指定するためには，コントロールのキャプションまたはコントロール ID を指定します。また，コントロールの種類（ボタン，リストボックスなど）も指定します。

次に，コントロールのキャプションを指定した AIT_CtrlSetFocus の使用例を示します。

図 2-16　AIT_CtrlSetFocus の使用例

```
if(AIT_FocusWindow("ｲﾝｽﾄｰﾙ先の選択", "#32770") > 0)
  AIT_CtrlSetFocus("次へ(&N) >",BUTTON_CTRL);
  AIT_Sleep(SLEEP_TIME_EVENTS);
  AIT_PlayKey("{ENTER}");
  AIT_Sleep(SLEEP_TIME_EVENTS);
  iLoopCount = 0;
Endif;
```

(3) AIT_VerifyExistence

AIT_VerifyExistence は，コントロールの存在を確認します。

例えば，同じキャプションを持つウィンドウが複数存在する場合，AIT_FocusWindow でウィンドウにフォーカスを設定しても，そのウィンドウが操作対象でないおそれがあります。このような場合に AIT_VerifyExistence を使用して，ウィンドウの中に特定のコントロールが存在することを確認し，そのウィンドウが操作対象であることを判定します。

コントロールを指定するためには，コントロールのキャプションまたはコントロール ID を指定します。また，コントロールの種類（ボタン，リストボックスなど）も指定します。戻り値は，コントロールが存在する場合は 1，存在しない場合は 0 になります。

次に，コントロールのキャプションを指定した AIT_VerifyExistence の使用例を示します。

図 2-17　AIT_VerifyExistence の使用例

```
if(AIT_FocusWindow("ｾｯﾄｱｯﾌﾟ", "#32770") > 0)
  if(AIT_VerifyExistence("終了 (&F)",BUTTON_CTRL)==1)
    AIT_CtrlSetFocus("終了 (&F)",BUTTON_CTRL);
    AIT_Sleep(SLEEP_TIME_EVENTS);
    AIT_PlayKey("{ENTER}");
    AIT_Sleep(SLEEP_TIME_EVENTS);
    iLoopCount = 0;
  Endif;
Endif;
```

(4) AIT_VerifyEnabled

AIT_VerifyEnabled は，コントロールが使用可能かどうかを確認します。

例えば，チェックボックスとボタンが関連づけられていて，チェックボックスをチェックするとボタンが活性化し（使用可能），チェックを外すとボタンが非活性（使用不可）になる場合があります。このような場合は，AIT_VerifyEnabled でボタンが使用可能かどうかを確認し，使用可能であれば「ボタンを押す」というユーザ操作をシミュレートします。

また，AIT_VerifyEnabled は，AIT_VerifyExistence や AIT_VerifyPos と組み合わせて使用します。AIT_VerifyExistence や AIT_VerifyPos でコントロールの存在を確認したあと，そのコントロールが使用可能かどうかを確認します。

コントロールを指定するためには，コントロールのキャプションまたはコントロール ID を指定します。また，コントロールの種類（ボタン，リストボックスなど）も指定します。戻り値は，コントロールが使用可能な場合は 1，使用できない場合は 0 になります。

次に，コントロールのキャプションを指定した AIT_VerifyEnabled の使用例を示します。

図 2-18　AIT_VerifyEnabled の使用例

```
if(AIT_FocusWindow("ｾｯﾄｱｯﾌﾟ", "#32770") > 0)
    if ((AIT_VerifyExistence("終了 (&F)",BUTTON_CTRL)==1)
    && ((AIT_VerifyEnabled("終了 (&F)",BUTTON_CTRL)==1)
        AIT_CtrlSetFocus("終了 (&F)",BUTTON_CTRL);
        AIT_Sleep(SLEEP_TIME_EVENTS);
        AIT_PlayKey("{ENTER}");
        AIT_Sleep(SLEEP_TIME_EVENTS);
        iLoopCount = 0;
    Endif;
Endif;
```

(5) AIT_VerifyPos

AIT_VerifyPos は，コントロールのタブオーダーを確認します。AIT_VerifyPos は AIT_VerifyExistence と同様の目的に使用しますが，ウィンドウ中に同じキャプションを持つコントロールが複数存在する場合に，タブオーダーも指定することによって，対象のコントロールの存在を確認します。

コントロールを指定するためには，コントロールのキャプションまたはコントロール ID を指定します。また，コントロールの種類（ボタン，リストボックスなど）とタブオーダーも指定します。戻り値は，コントロールのタブオーダーが指定したパラメタと一致した場合は 1，一致しない場合は 0 になります。

次に，コントロールのキャプションを指定した AIT_VerifyPos の使用例を示します。

図 2-19　AIT_VerifyPos の使用例

```
if(AIT_FocusWindow("ｾｯﾄｱｯﾌﾟ", "#32770") > 0)
    if(AIT_VerifyPos("終了 (&F)",BUTTON_CTRL,6)==1)
        AIT_CtrlSetFocus("終了 (&F)",BUTTON_CTRL);
        AIT_Sleep(SLEEP_TIME_EVENTS);
        AIT_PlayKey("{ENTER}");
        AIT_Sleep(SLEEP_TIME_EVENTS);
        iLoopCount = 0;
    Endif;
Endif;
```

(6) AIT_PlayKey

AIT_PlayKey は，キーボード操作をシミュレートします。

パラメタとして，キーボードから入力する文字列を指定します。例えば，パラメタとして「abcd」を指定すると，「abcd」という文字列が入力されます。また，［Esc］キーの入力を意味する {ESC} や［Alt］＋［F］キーの入力を意味する %(F) など，特殊な意味を持つ文字を入力することもできます。

次に，AIT_PlayKey の使用例を示します。

図 2-20　AIT_PlayKey の使用例

```
        if(AIT_FocusWindow("ｾｯﾄｱｯﾌﾟ", "#32770") > 0)
            if(AIT_VerifyPos("ｱｶｳﾝﾄ(&T)",BUTTON_CTRL,6)==1)
              AIT_PlayKey("administrator");
              AIT_Sleep(SLEEP_TIME_EVENTS);
              AIT_PlayKey("{ENTER}");
              iLoopCount = 0;
            Endif;
        Endif;
```

(7) 自動生成されたウィンドウ処理の例

レコーディングして自動生成された AIT ファイルには，ウィンドウの情報やウィンドウに対する処理が記録されています。自動生成されたウィンドウ処理の例を次の図に示します。

図 2-21　自動生成されたウィンドウ処理の例

```
. . .
MAIN
{
    AIT_SetDefaultWaitTimeout(1.0);
    AIT_DMPSTRC();
    DM_RTN = NG_END;
    iCapsLockState = AIT_GetKeyState(CAPSLOCK);
    iNumLockState = AIT_GetKeyState(NUMLOCK);
    iScrollLockState = AIT_GetKeyState(SCROLLLOCK);
    iInsertLockState = AIT_GetKeyState(INSERTLOCK);
    bRtn= AIT_Exec(InstallerName,SW_SHOWNORMAL);
    if(bRtn == false)
        iLoopCount = iLoopMax;
    Endif;
    while(iLoopCount < iLoopMax)
        if(AIT_FocusWindow("Acrobat Readerを解凍中です...", "#32770") > 0)
            AIT_Sleep(SLEEP_TIME);
            iLoopCount=0;
        Endif;
        if(AIT_FocusWindow("Acrobat Reader 5.0.5 のセットアップ", "#32770") > 0)
            AIT_CtrlSetFocus("次へ(&N) >",BUTTON_CTRL);
            AIT_Sleep(SLEEP_TIME_EVENTS);
            AIT_PlayKey("{ALT_LOCK}n{ALT_UNLOCK}");
            AIT_Sleep(SLEEP_TIME_EVENTS);
            iLoopCount=0;
        Endif;
. . .
```

ウィンドウの検索

コントロールにフォーカスを当てる

[Alt] + [n] キーを押す

2.6.2　自動生成されたフラグについて

レコーディング中に，同じキャプションを持つ複数のウィンドウが処理される場合，生成された AIT ファイルでは一つのウィンドウの処理として記録されます。生成された AIT ファイルでは，次に示すフラグを利用して，各ウィンドウの処理を記述しています。

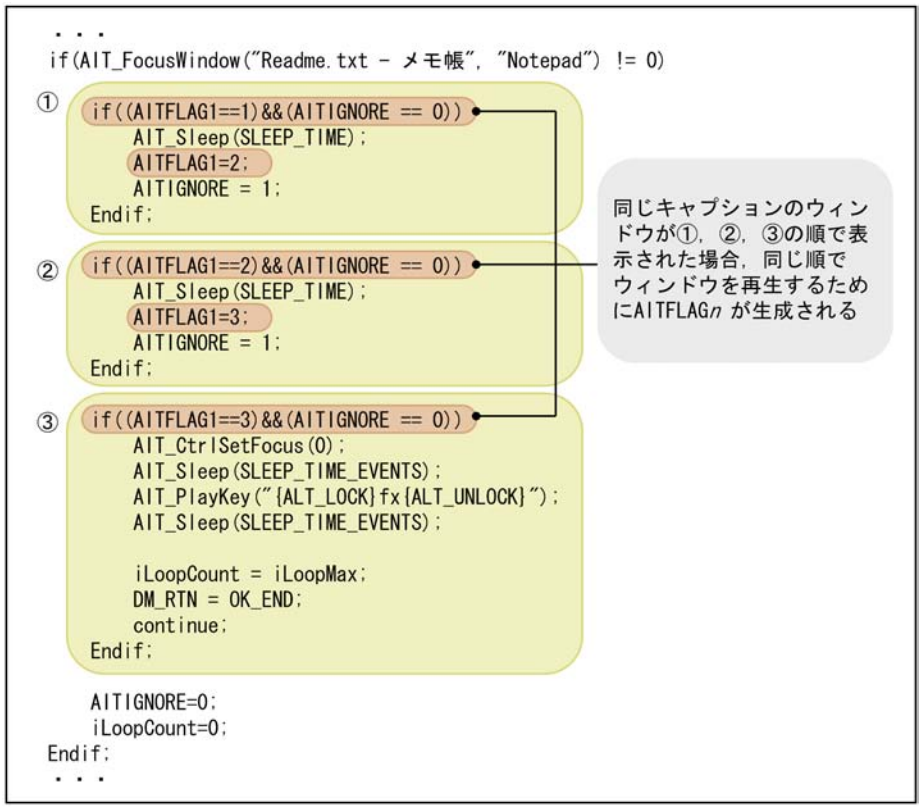
- ウィンドウフラグ（AITFLAG1 ～ *n*）
- イベントフラグ（AITEVENTFLAG1 ～ *n*）
- 無効フラグ（AITIGNORE）

それぞれについて説明します。

(1) ウィンドウフラグ（AITFLAG1 ～ *n*）

ウィンドウフラグは，ウィンドウが表示される順序を指定します。ウィンドウフラグの生成例を次に示します。

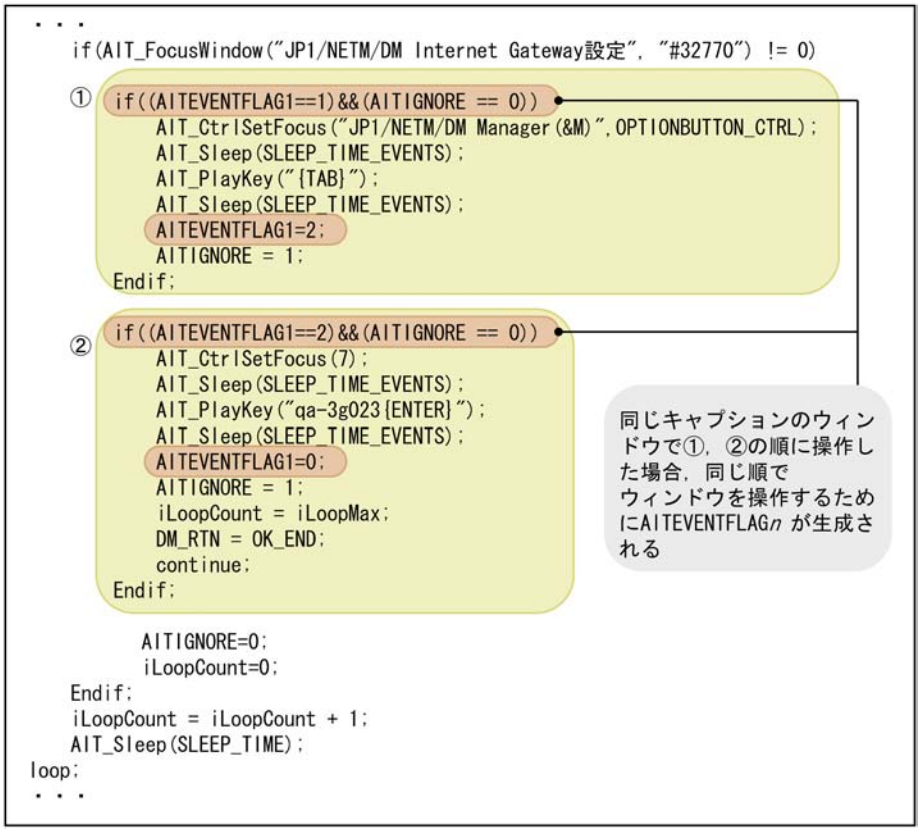
図 2-22 ウィンドウフラグの生成例



(2) イベントフラグ（AITEVENTFLAG1 ～ *n*）

イベントフラグは，ウィンドウ操作の順序を指定します。イベントフラグの生成例を次に示します。

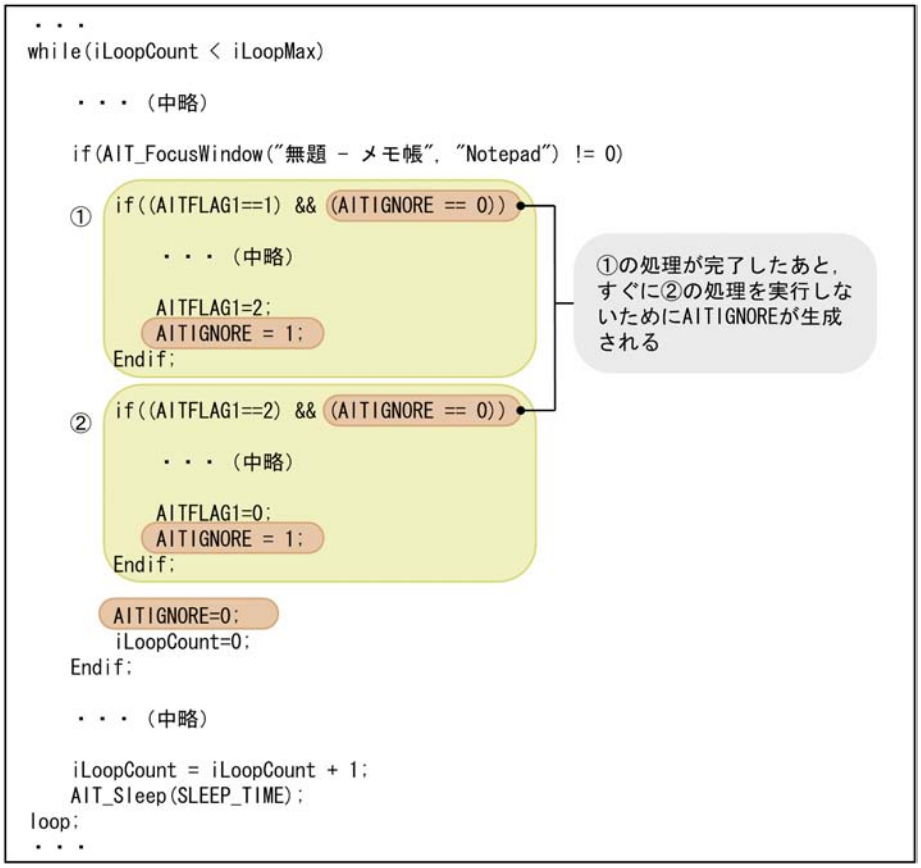
図 2-23　イベントフラグの生成例



(3) 無効フラグ（AITIGNORE）

無効フラグは，ある一つのウィンドウが閉じる前に次の操作が実行されることを防ぎます。無効フラグの生成例を次に示します。

図 2-24　無効フラグの生成例



2.6.3　自動生成された AIT ファイルの確認と修正のポイント

レコーダで AIT ファイルを自動生成したあと，次の事項に注意しながら，生成されたコードを確認，修正してください。

(1) インストール条件を変えてレコーディングを繰り返す

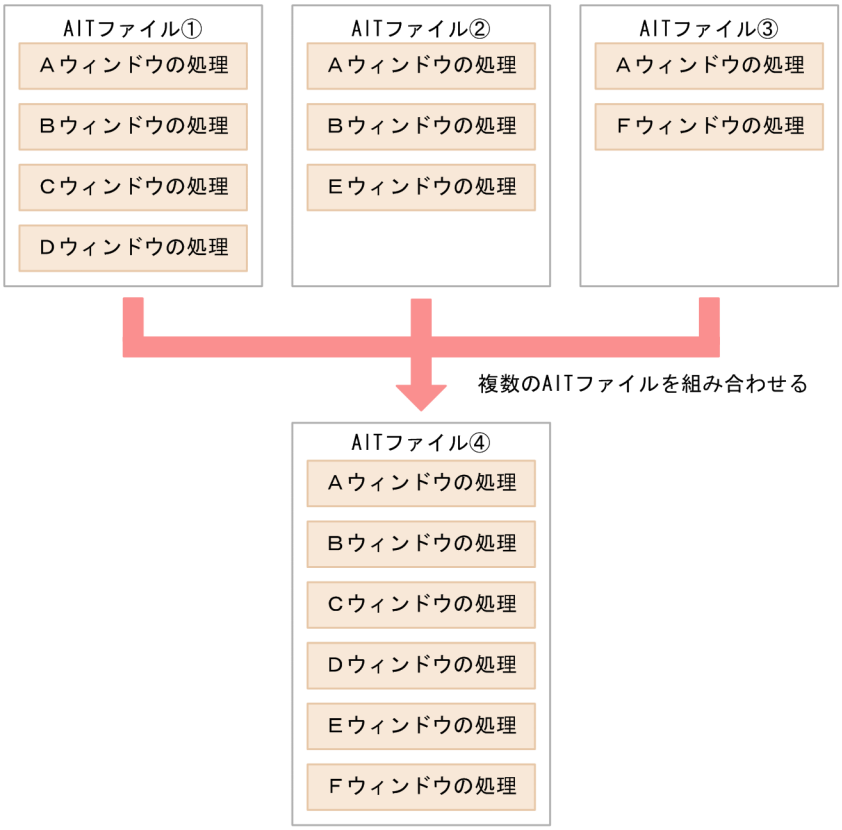
ハードディスクの空き容量や OS などのインストール条件に応じて，インストーラは異なるウィンドウを出力します。これらすべてのウィンドウ出力，すべての事象を 1 回のレコーディングで記録することはできません。すべてのウィンドウに対する操作をシミュレートするためには，インストール条件を変えてレコーディングを繰り返し，複数の AIT ファイルを生成する必要があります。インストール条件としては，次のような条件が考えられます。

インストール条件	説明
OS	OS が異なると，インストーラの出力するウィンドウが異なる場合があります。
PC 環境	ハードディスクの空き容量不足，前提プログラムのインストール有無，新規インストールか上書きインストールかなど，インストール対象の PC 環境によって，インストーラの出力するウィンドウが異なる場合があります。
ユーザ操作	インストール先ディレクトリの変更や，インストール操作のキャンセルなど，ユーザ操作が異なると，インストーラの出力するウィンドウも異なります。

インストール条件ごとに複数の AIT ファイルが自動生成されたら，それぞれの AIT ファイルから，必要なコードを抜き出して組み合わせます。必要なコードとは，インストール条件に応じて出力される，異なるウィンドウへの操作をシミュレートするコードです。

複数の AIT ファイルを組み合わせる場合の例を次の図に示します。

図 2-25　複数の AIT ファイルの組み合わせ例



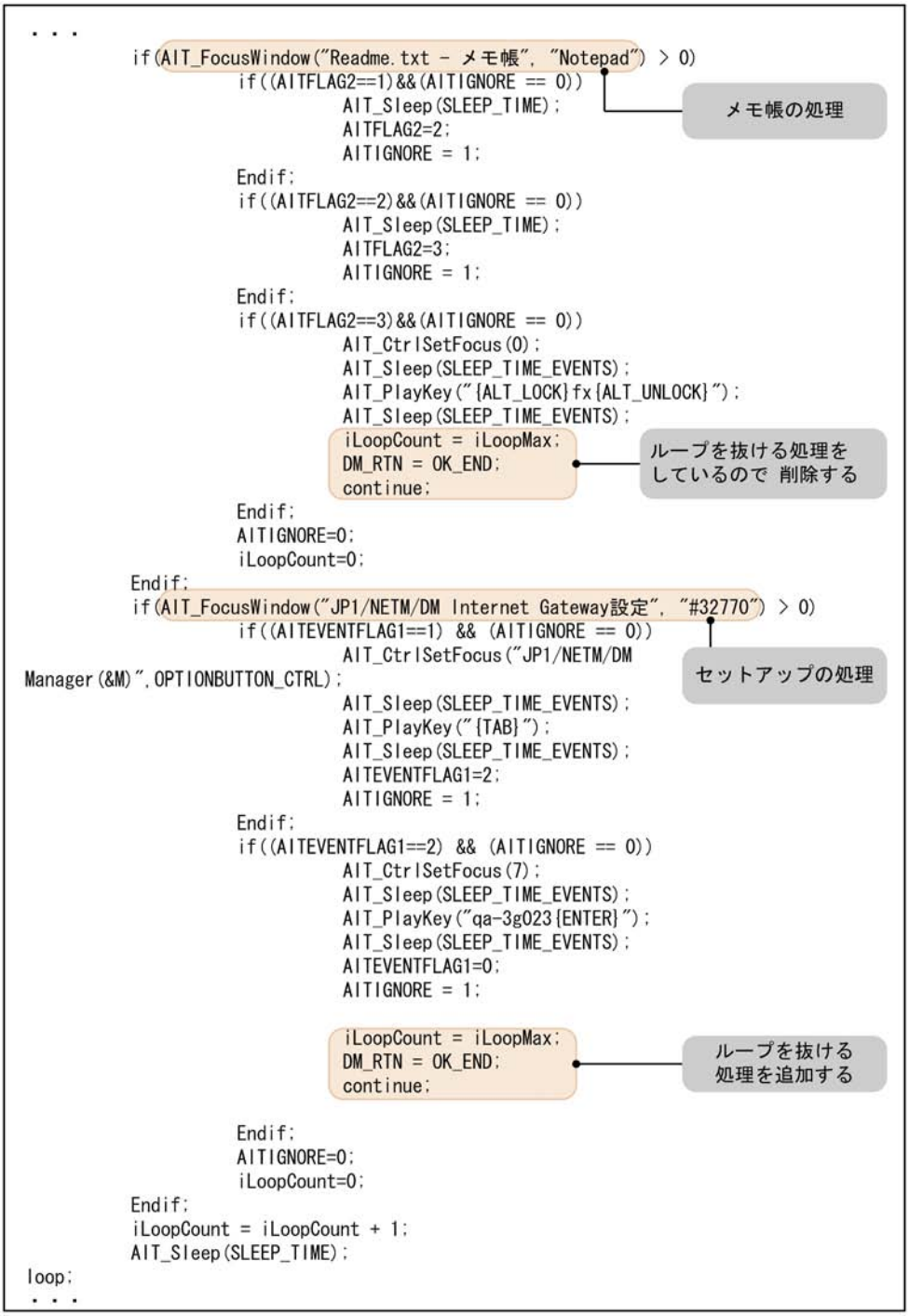
(2) インストール操作の終了を正しく判定する

レコーダは，最後にユーザ操作を行ったウィンドウに対して，インストール正常終了のウィンドウ処理を自動生成します。しかし，最後にユーザ操作を行ったウィンドウが，インストールの最後に出力されるウィンドウではないおそれもあります。

例えば，ユーザ操作の最後で Readme を表示するための「メモ帳」を閉じて，レコーディングを終了した場合，「メモ帳」の操作に対して，インストールの終了処理が自動生成されます。そのため，インストール途中で「メモ帳」が表示されると，実際にはインストール途中であっても，インストールが終了してしまいます。

したがって，レコーディング時には，インストールの最後に出力されるウィンドウで，ユーザ操作を終了するようにしてください。または，手作業で AIT ファイルを修正し，インストールが終了するまで「メモ帳」を閉じないように順序を制御してください。次に AIT ファイルの修正例を示します。

図 2-26　AIT ファイルの修正例




```
・・・
        if(AIT_FocusWindow("Readme.txt - メモ帳", "Notepad") > 0)
                if((AITFLAG2==1)&&(AITIGNORE == 0))
                        AIT_Sleep(SLEEP_TIME);
                        AITFLAG2=2;
                        AITIGNORE = 1;
                Endif;
                if((AITFLAG2==2)&&(AITIGNORE == 0))
                        AIT_Sleep(SLEEP_TIME);
                        AITFLAG2=3;
                        AITIGNORE = 1;
                Endif;
                if((AITFLAG2==3)&&(AITIGNORE == 0))
                        AIT_CtrlSetFocus(0);
                        AIT_Sleep(SLEEP_TIME_EVENTS);
                        AIT_PlayKey("{ALT_LOCK}fx{ALT_UNLOCK}");
                        AIT_Sleep(SLEEP_TIME_EVENTS);
                        iLoopCount = iLoopMax;
                        DM_RTN = OK_END;
                        continue;
                Endif;
                AITIGNORE=0;
                iLoopCount=0;
        Endif;
        if(AIT_FocusWindow("JP1/NETM/DM Internet Gateway設定", "#32770") > 0)
                if((AITEVENTFLAG1==1) && (AITIGNORE == 0))
                        AIT_CtrlSetFocus("JP1/NETM/DM
Manager(&M)",OPTIONBUTTON_CTRL);
                        AIT_Sleep(SLEEP_TIME_EVENTS);
                        AIT_PlayKey("{TAB}");
                        AIT_Sleep(SLEEP_TIME_EVENTS);
                        AITEVENTFLAG1=2;
                        AITIGNORE = 1;
                Endif;
                if((AITEVENTFLAG1==2) && (AITIGNORE == 0))
                        AIT_CtrlSetFocus(7);
                        AIT_Sleep(SLEEP_TIME_EVENTS);
                        AIT_PlayKey("qa-3g023{ENTER}");
                        AIT_Sleep(SLEEP_TIME_EVENTS);
                        AITEVENTFLAG1=0;
                        AITIGNORE = 1;

                        iLoopCount = iLoopMax;
                        DM_RTN = OK_END;
                        continue;

                Endif;
                AITIGNORE=0;
                iLoopCount=0;
        Endif;
        iLoopCount = iLoopCount + 1;
        AIT_Sleep(SLEEP_TIME);
loop;
・・・
```

また，同じキャプションのダイアログボックスが，インストールの途中と最後に出力される場合，インストール操作の終了判定が正しく行われるように，異なるラベルやボタンを記述して，二つのダイアログボックスを区別してください。

(3) 変化するテキストをウィンドウの判定に使わない

ウィンドウを判定するために，ウィンドウテキストだけでなく，ウィンドウ内のコントロールのテキストを判定条件として使用できます。このような場合，変化するテキストを判定条件に使用しないでください。

例えば，インストール時に PC の空きディスク容量が，「空きディスク容量：2252195 K」のように表示されることがあります。空きディスク容量はインストールする PC によって値が異なるので，「2252195 K」は，ウィンドウを判定する条件として使用できません。

自動生成された AIT ファイルに，PC 環境や OS によって変化するテキストが判定条件として使用されている場合は，削除してください。または，テキストの変化しない部分（例えば「空きディスク容量」）だけを判定条件として使用してください。

(4) 無関係なウィンドウに対するコードは削除する

レコーディング時に，インストーラとは別のアプリケーションが起動していた場合，インストーラとは無関係なウィンドウに対するコードが生成されているおそれがあります。このような場合には，無関係なウィンドウに対するコードは削除してください。

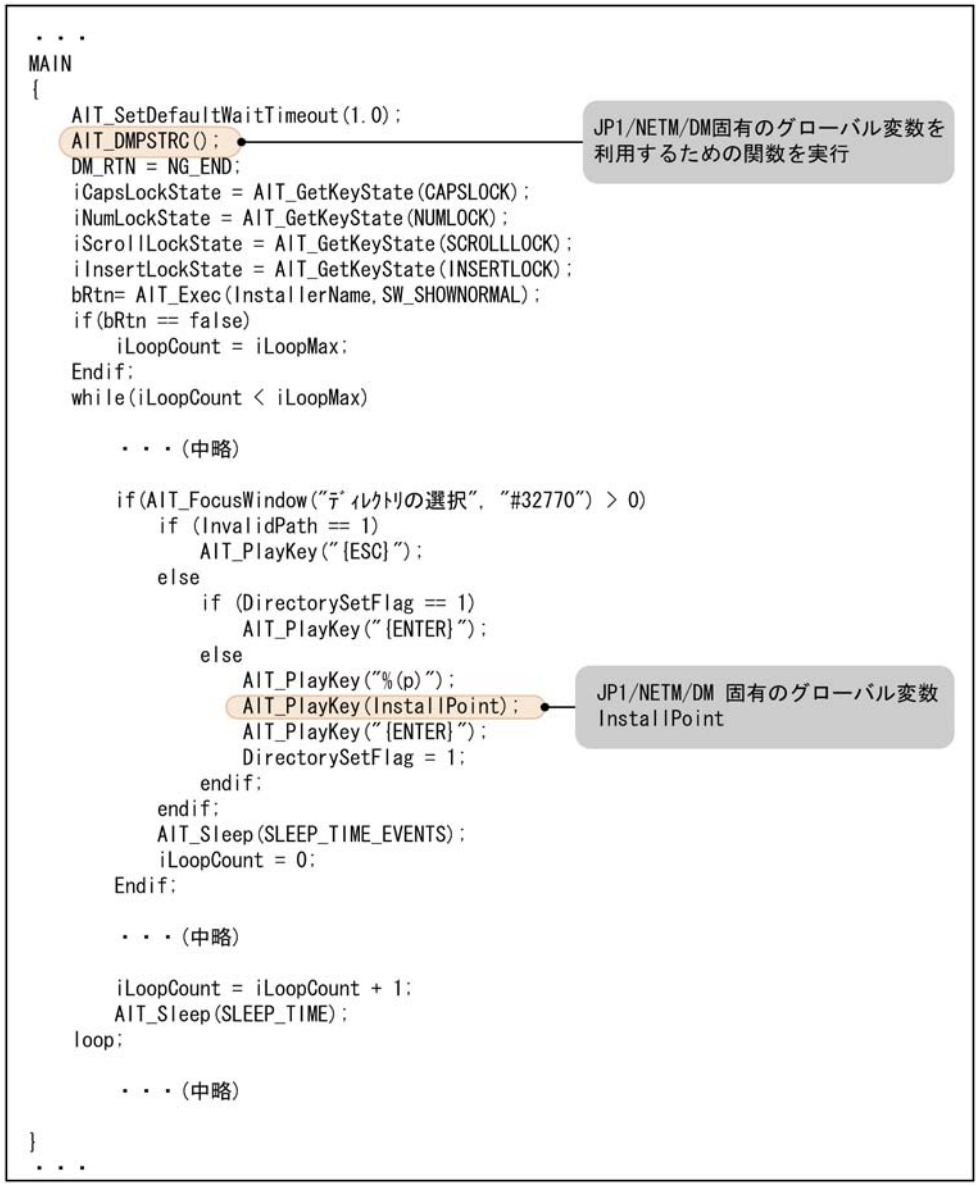
2.6.4 パッケージャおよびリモートインストールマネージャとの連動

パッケージング時またはリモートインストール時に，「インストール先ディレクトリ」や「所有者名」などのパッケージ内容を変更した場合，AIT ファイルの PACKAGE_INFO セクションは自動的に書き換えられますが，MAIN セクションの関連部分は書き換えられません。そのため，必要に応じて，JP1/NETM/DM 固有のグローバル変数を利用してください。使用できる JP1/NETM/DM 固有のグローバル変数を示します。

JP1/NETM/DM 固有のグローバル変数	説明
InstallerName	PACKAGE_INFO セクションで指定したインストールプログラム名です。この値は，パッケージング時またはリモートインストール時には変更できません。 なお，リモートインストール時は，クライアントのインストールワークディレクトリのパスにインストールプログラム名を結合した値となります。そのため，リモートインストール時と AIT ファイルのデバッグ時で値が異なります。
InstallDrive	パッケージング時またはリモートインストール時に指定した，インストール先ドライブ名です。デフォルトは，PACKAGE_INFO セクションで指定したインストール先ドライブ名です。
InstallDirectory	パッケージング時またはリモートインストール時に指定した，インストール先ディレクトリ名です。デフォルトは，PACKAGE_INFO セクションで指定したインストール先ディレクトリ名です。
InstallPoint	InstallDrive 変数と InstallDirectory 変数の内容を結合したインストール位置です。つまり，ドライブ名を含むインストール先ディレクトリです。
EUser	パッケージング時またはリモートインストール時に指定した，所有者名です。英語を入力したら EUser が，日本語を入力したら JUser が使用できます。
JUser	
ECompany	パッケージング時またはリモートインストール時に指定した，会社名です。英語を入力したら ECompany が，日本語を入力したら JCompany が使用できます。
JCompany	
SerialNumber	パッケージング時またはリモートインストール時に指定した，シリアルナンバーです。ライセンスキーの入力などに使用します。
IconGroupName	パッケージング時またはリモートインストール時に指定した，アイコングループです。アイコングループの変更などで使用します。

グローバル変数の利用例を次の図に示します。

図 2-27　グローバル変数の利用例



2.6.5　エラー処理の追加とリターンコードの設定

異常時にインストーラが出力するウィンドウにも自動応答するように，エラー処理を追加します。インストール時に異常が発生した場合，インストールを続行するか，それともインストールを中断して異常終了とするかを決めて，それに従ったウィンドウ処理をします。

また，インストールが正常に終了したのか，異常終了したのかがわかるようにリターンコードを設定します。リターンコードを設定しておくと，配布管理システムでエラーの内容を確認できます。リターンコードは，ジョブの［詳細情報］ダイアログボックスに表示される保守コードの，左から 9 番目と 10 番目に表示される 2 けたの値です。

異常発生時のエラー処理とリターンコードの設定例を次の図に示します。

図 2-28　エラー処理の追加とリターンコードの設定例（1/2）

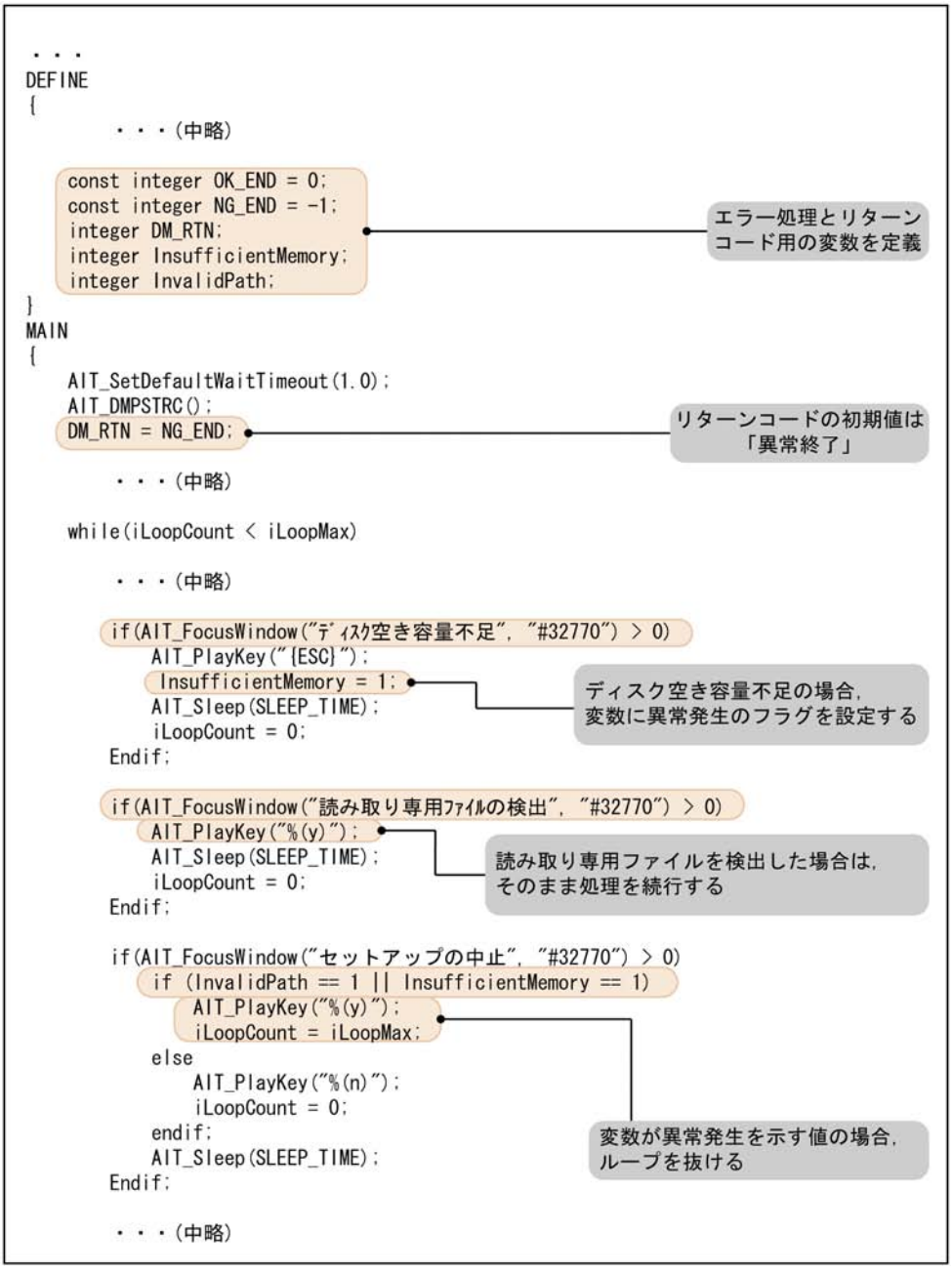
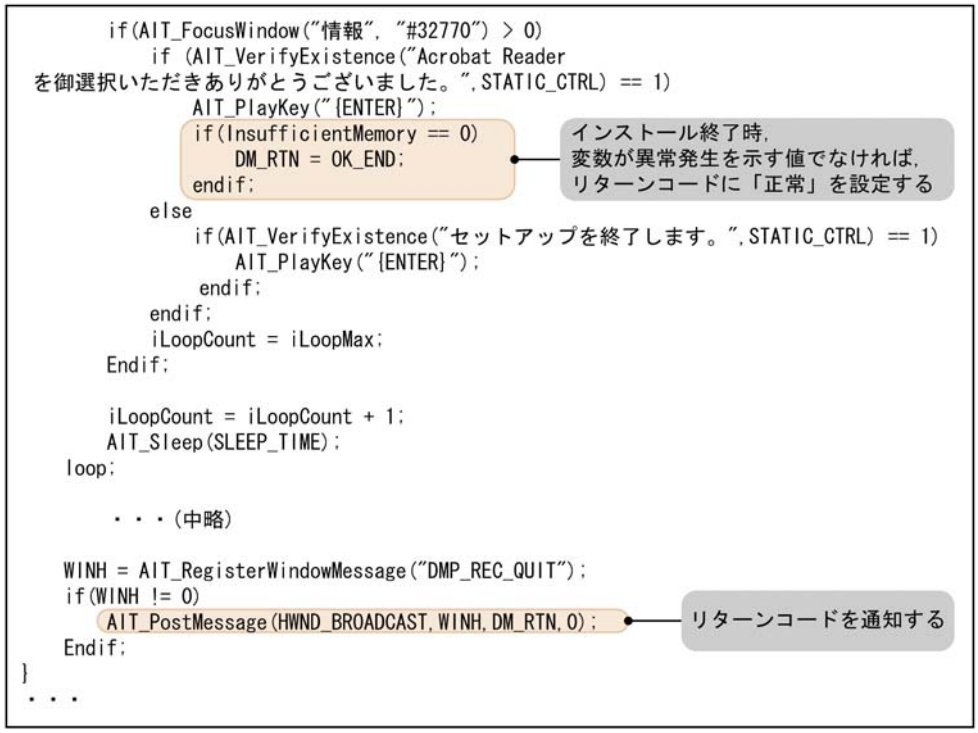


図 2-29　エラー処理の追加とリターンコードの設定例（2/2）




```
        if(AIT_FocusWindow("情報", "#32770") > 0)
            if (AIT_VerifyExistence("Acrobat Reader
  を御選択いただきありがとうございました。",STATIC_CTRL) == 1)
                AIT_PlayKey("{ENTER}");
                if(InsufficientMemory == 0)
                    DM_RTN = OK_END;
                endif;
            else
                if(AIT_VerifyExistence("セットアップを終了します。",STATIC_CTRL) == 1)
                    AIT_PlayKey("{ENTER}");
                endif;
            endif;
            iLoopCount = iLoopMax;
        Endif;

        iLoopCount = iLoopCount + 1;
        AIT_Sleep(SLEEP_TIME);
    loop;

        ・・・(中略)

    WINH = AIT_RegisterWindowMessage("DMP_REC_QUIT");
    if(WINH != 0)
        AIT_PostMessage(HWND_BROADCAST,WINH,DM_RTN,0);
    Endif;
}
・・・
```

2.6.6　AIT ファイルの完成例

Acrobat Reader 5.05 をリモートインストールする場合の，AIT ファイルの完成例を示します。なお，JP1/NETM/DM では，Acrobat Reader 5.05 に対応する AIT ファイルを標準で提供しています。

```
PACKAGE_INFO
{
  PackageID         = "ADOBEACROBATREADER";
  Product           = "Adobe Acrobat Reader 5.05";
  Version           = "505";
  InstallerName     = "Ar505jpn.exe";
  InstallDrive      = "C:";
  InstallDirectory  = "'¥Program Files'¥Adobe'¥Acrobat 5.0";
}
DEFINE
{
  integer iLoopCount = 0;
  integer iLoopMax = 60;
  integer DM_RTN;
  integer WINH;
  bool bRtn;
  const integer OK_END = 0;
  const integer NG_END = -1;
  float SLEEP_TIME = 1.0;
  float SLEEP_TIME_RESTART = 10.0;
  float SLEEP_TIME_EVENTS = 0.5;
  integer InsufficientMemory;
  integer InvalidPath;
  integer DirectorySetFlag;
}
MAIN
{
  AIT_SetDefaultWaitTimeout(1.0);
  AIT_DMPSTRC();
  DM_RTN = NG_END;
  bRtn= AIT_Exec(InstallerName,SW_SHOWNORMAL);
```

```
if(bRtn == false)
        iLoopCount = iLoopMax;
Endif;
while(iLoopCount < iLoopMax)
    if(AIT_FocusWindow("Acrobat Readerを解凍中です...", "#32770") > 0)
        AIT_Sleep(SLEEP_TIME);
        iLoopCount = 0;
    Endif;

    if(AIT_FocusWindow("Unpacking Acrobat Reader...", "#32770") > 0)
        AIT_Sleep(SLEEP_TIME);
        iLoopCount = 0;
    Endif;

    if(AIT_FocusWindow("セットアップの中止", "#32770") > 0)
        if (InvalidPath == 1 || InsufficientMemory == 1)
            AIT_PlayKey("%(y)");
            iLoopCount = iLoopMax;
        else
            AIT_PlayKey("%(n)");
            iLoopCount = 0;
        endif;
        AIT_Sleep(SLEEP_TIME);
    Endif;

    if(AIT_FocusWindow("上書き確認", "#32770") > 0)
        AIT_PlayKey("%(e)");
        AIT_Sleep(SLEEP_TIME);
        iLoopCount = 0;
    Endif;

    if(AIT_FocusWindow("セットアップ初期化エラー", "#32770") > 0)
        AIT_PlayKey("{ENTER}");
        AIT_Sleep(SLEEP_TIME);
        iLoopCount = iLoopMax;
    Endif;

    if(AIT_FocusWindow("PackageForTheWeb", "#32770") > 0)
        if (InsufficientMemory == 1)
            AIT_PlayKey("%(y)");
            iLoopCount = iLoopMax;
        else
            AIT_PlayKey("%(n)");
            iLoopCount = 0;
        endif;
        AIT_Sleep(SLEEP_TIME);
    Endif;

    if(AIT_FocusWindow("Error", "#32770") > 0)
        AIT_PlayKey("{ENTER}");
        AIT_Sleep(SLEEP_TIME);
        iLoopCount = 0;
    Endif;

    if(AIT_FocusWindow("セットアップの完了", "#32770") > 0)
        AIT_PlayKey("{DOWN}");
        AIT_PlayKey("{ENTER}");
        AIT_Sleep(SLEEP_TIME);
        iLoopCount = iLoopMax;
        DM_RTN = OK_END;
    Endif;

    if(AIT_FocusWindow("PackageForTheWeb Stub", "#32770") > 0)
        AIT_PlayKey("%(C)");
        InsufficientMemory = 1;
        AIT_Sleep(SLEEP_TIME);
        iLoopCount = iLoopMax;
    Endif;

    if(AIT_FocusWindow("Ar505jpn", "#32770") > 0)
        AIT_PlayKey("%(C)");
        InsufficientMemory = 1;
        AIT_Sleep(SLEEP_TIME);
        iLoopCount = iLoopMax;
```

```
Endif;

if(AIT_FocusWindow("_ins5576", "#32770") > 0)
    AIT_PlayKey("%(C)");
    AIT_Sleep(SLEEP_TIME);
    iLoopCount = iLoopMax;
Endif;

if(AIT_FocusWindow("PackageForTheWeb Error", "#32770") > 0)
    AIT_PlayKey("{ENTER}");
    InsufficientMemory = 1;
    AIT_Sleep(SLEEP_TIME);
    iLoopCount = iLoopMax;
Endif;

if(AIT_FocusWindow("PackageForTheWeb エラー", "#32770") > 0)
    AIT_PlayKey("{ENTER}");
    InsufficientMemory = 1;
    AIT_Sleep(SLEEP_TIME);
    iLoopCount = iLoopMax;
Endif;

if(AIT_FocusWindow("+TEMP", "#32770") > 0)
    AIT_PlayKey("{ENTER}");
    AIT_Sleep(SLEEP_TIME);
    iLoopCount = 0;
Endif;

if(AIT_FocusWindow("エラー", "#32770") > 0)
    if (AIT_VerifyExistence("中止(&A)",BUTTON_CTRL) > 0)
        AIT_PlayKey("%(a)");
        iLoopCount = iLoopMax;
    else
        if (AIT_VerifyExistence("OK",BUTTON_CTRL) > 0)
            AIT_PlayKey("{ENTER}");
            InsufficientMemory = 1;
            iLoopCount = 0;
        endif;
    endif;
    AIT_Sleep(SLEEP_TIME);
Endif;

if(AIT_FocusWindow("警告", "#32770") > 0)
    AIT_PlayKey("{ENTER}");
    AIT_Sleep(SLEEP_TIME);
    iLoopCount = iLoopMax;
Endif;

if(AIT_FocusWindow("質問", "#32770") > 0)
    AIT_PlayKey("%(n)");
    AIT_Sleep(SLEEP_TIME);
    iLoopCount = 0;
Endif;

if(AIT_FocusWindow("ComponentMoveData Error Information", "#32770") > 0)
    AIT_PlayKey("{ENTER}");
    AIT_Sleep(SLEEP_TIME);
    iLoopCount = 0;
Endif;

if(AIT_FocusWindow("Acrobat Reader のセットアップ", "#32770") > 0)
    AIT_PlayKey("{ENTER}");
    AIT_Sleep(SLEEP_TIME);
    iLoopCount = iLoopMax;
Endif;

if(AIT_FocusWindow("ディスク空き容量不足", "#32770") > 0)
    AIT_PlayKey("{ESC}");
    InsufficientMemory = 1;
    AIT_Sleep(SLEEP_TIME);
    iLoopCount = 0;
Endif;
```

```
        if(AIT_FocusWindow("読み取り専用ファイルの検出", "#32770") > 0)
            AIT_PlayKey("%(y)");
            AIT_Sleep(SLEEP_TIME);
            iLoopCount = 0;
        Endif;

        if(AIT_FocusWindow("Acrobat Reader", "#32770") > 0)
            InsufficientMemory = 1;
            if (AIT_VerifyExistence("再試行(&R)",BUTTON_CTRL) > 0)
                AIT_PlayKey("{ESC}");
                iLoopCount = iLoopMax;
            else
                if (AIT_VerifyExistence("OK",BUTTON_CTRL) > 0)
                    AIT_PlayKey("{ENTER}");
                    iLoopCount = iLoopMax;
                endif;
            endif;
            AIT_Sleep(SLEEP_TIME);
        Endif;

        if(AIT_FocusWindow("セットアップ", "#32770") > 0)
            if(AIT_VerifyExistence("~ディレクトリ",STATIC_CTRL) > 0)
                AIT_PlayKey("%(y)");
                AIT_Sleep(SLEEP_TIME);
            else
                if(AIT_VerifyExistence("~指定されたディレクトリ",STATIC_CTRL) > 0)
                    InvalidPath = 1;
                    AIT_PlayKey("{ENTER}");
                    AIT_Sleep(SLEEP_TIME);
                    iLoopCount = 0;
                endif;
            endif;
            iLoopCount = 0;
        Endif;

        if(AIT_FocusWindow("Acrobat Reader 5.0.5 のセットアップ", "#32770") > 0)
            AIT_PlayKey("%(n)");
            AIT_Sleep(SLEEP_TIME_EVENTS);
            iLoopCount = 0;
        Endif;

        if(AIT_FocusWindow("インストール先の選択", "#32770") > 0)
            if ( (AIT_FindSubStr(InstallPoint,"%",0) != -1) ||
(AIT_FindSubStr(InstallPoint,"~",0) != -1) )
                InvalidPath = 1;
                DirectorySetFlag = 1;
            endif;
            if (DirectorySetFlag == 0)
                AIT_PlayKey("%(r)");
            else
                if (InvalidPath == 1 || InsufficientMemory == 1)
                    AIT_PlayKey("{ESC}");
                else
                    AIT_PlayKey("%(n)");
                endif;
            endif;
            AIT_Sleep(SLEEP_TIME_EVENTS);
            iLoopCount = 0;
        Endif;

        if(AIT_FocusWindow("ディレクトリの選択", "#32770") > 0)
            if (InvalidPath == 1)
                AIT_PlayKey("{ESC}");
            else
                if (DirectorySetFlag == 1)
                    AIT_PlayKey("{ENTER}");
                else
                    AIT_PlayKey("%(p)");
                    AIT_PlayKey(InstallPoint);
                    AIT_PlayKey("{ENTER}");
                    DirectorySetFlag = 1;
                endif;
            endif;
```

```
            AIT_Sleep(SLEEP_TIME_EVENTS);
            iLoopCount = 0;
        Endif;

        if(AIT_FocusWindow("情報", "#32770") > 0)
            if (AIT_VerifyExistence("Acrobat Reader を御選択いただきありがとうございまし
た。",STATIC_CTRL) == 1)
                AIT_PlayKey("{ENTER}");
                if(InsufficientMemory == 0)
                    DM_RTN = OK_END;
                endif;
            else
                if(AIT_VerifyExistence("セットアップを終了します。",STATIC_CTRL) == 1)
                    AIT_PlayKey("{ENTER}");
                endif;
            endif;
            iLoopCount = iLoopMax;
        Endif;

        if (AIT_FocusWindow("", "#32770") > 0)
            if(AIT_VerifyExistence("はい(&Y)",BUTTON_CTRL) > 0)
                AIT_PlayKey("%(y)");
                iLoopCount = iLoopMax;
            endif;
            AIT_Sleep(SLEEP_TIME);
        endif;

        if (AIT_FocusWindow("", "#32770") > 0)
            if(AIT_VerifyExistence("OK",BUTTON_CTRL) > 0)
                AIT_PlayKey("{ENTER}");
                iLoopCount = iLoopMax;
            endif;
            AIT_Sleep(SLEEP_TIME);
        endif;

       if(AIT_FocusWindow("Acrobat Reader のセットアップ", "InstallShield_Win") > 0)
            AIT_Sleep(SLEEP_TIME_EVENTS);
            iLoopCount=0;
        Endif;

        iLoopCount = iLoopCount + 1;
        AIT_Sleep(SLEEP_TIME);
    loop;
    WINH = AIT_RegisterWindowMessage("DMP_REC_QUIT");
    if(WINH != 0)
        AIT_PostMessage(HWND_BROADCAST,WINH,DM_RTN,0);
    Endif;
}
ERROR
{
    DM_RTN = NG_END;
    WINH = AIT_RegisterWindowMessage("DMP_REC_QUIT");
    if(WINH != 0)
        AIT_PostMessage(HWND_BROADCAST,WINH,DM_RTN,0);
    Endif;
}
```

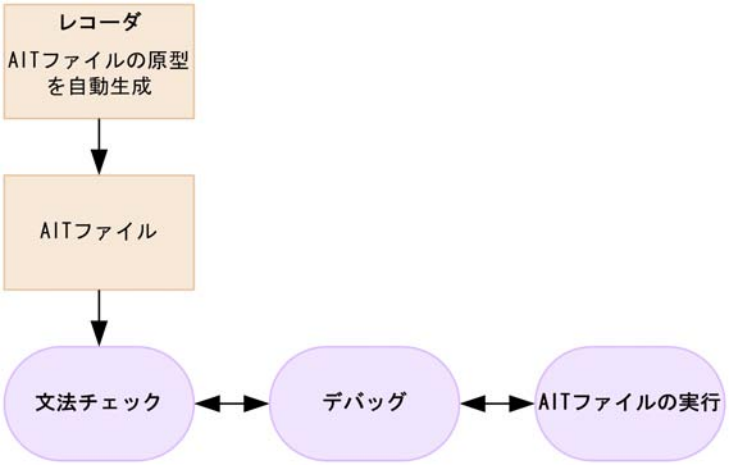
2.7　AIT ファイルをデバッグする

一とおり AIT ファイルの作成が終わったら，文法チェック，実行，デバッグを繰り返し，ユーザ操作を正しくシミュレートする AIT ファイルを完成させます。AIT ファイルを実行するために，コンパイルする必要はありません。

なお，Windows 8・Windows Server 2012・Windows 7・Windows Server 2008・Windows Vista 版 JP1/NETM/DMClient で Administrator 権限を必要とするプログラムの AIT ファイルをデバッグする場合，Administrator 権限で［Automatic Installation Tool］ウィンドウを起動してください。プログラムが一般ユーザ権限で実行できる場合は，Administrator 権限および一般ユーザ権限のどちらからでもデバッグを実行できます。

AIT ファイルのデバッグの流れを次の図に示します。

図 2-30　AIT ファイルのデバッグの流れ



AIT ファイルを開くと，コーディングのバグを修正するための［ビルド］メニューと［デバッグ］メニューが使用可能になります。

［ビルド］メニューは，AIT ファイルの文法上の誤りを検出したり，アクティブな AIT ファイルを実行したりできます。

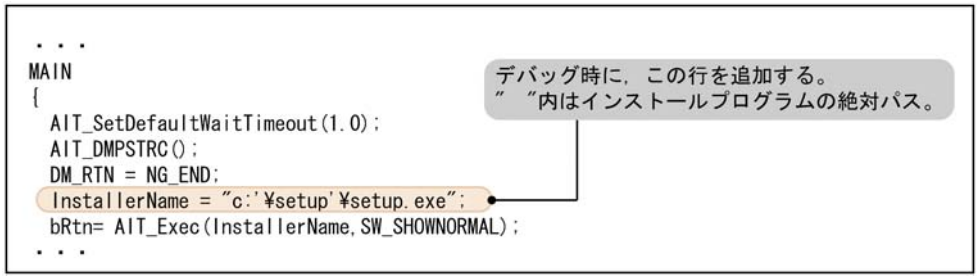
［デバッグ］メニューは，ブレークポイントを設定することによって，AIT ファイルを指定した位置まで実行できます。また，監視ウィンドウで変数の値を参照したり，更新したりできます。

これらの機能は，拡張子 .ais を持つ AIT ファイルにだけ有効です。

デバッグ時の注意事項

PACKAGE_INFO セクションの InstallerName には，インストールプログラム名が，パッケージングするディレクトリからの相対パスで記述されています。デバッグ時には，一時的に MAIN セクションで，これを絶対パスに置き換える必要があります。デバッグが完了したら，元に戻しておいてください。ただし，［文法チェック］のときは，絶対パスに置き換える必要はありません。次に，InstallerName を絶対パスに置き換える例を示します。

図 2-31 InstallerName を絶対パスに置き換える例



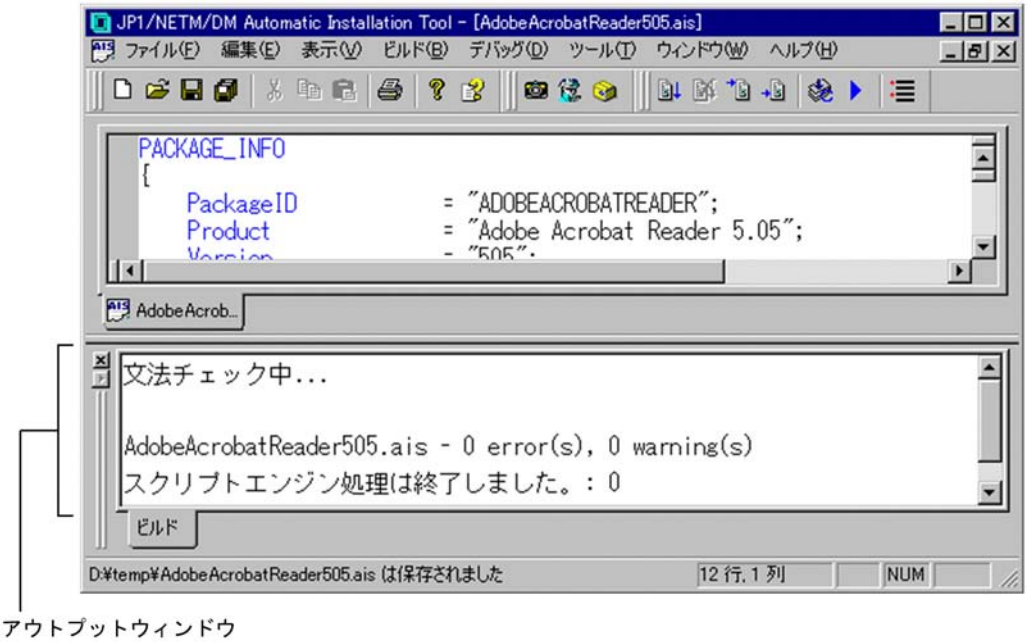
2.7.1 文法チェックと実行

Automatic Installation Tool では，次の［ビルド］メニューを使用して，AIT ファイルの文法チェックと実行ができます。

● 文法チェック
アクティブな AIT ファイルの文法をチェックし，アウトプットウィンドウに文法エラーと警告を表示します。

● 実行
アクティブな AIT ファイルの文法をチェックしたあと，エラーがなければ，AIT ファイルを実行します。AIT ファイルの実行が完了すると，終了コードがアウトプットウィンドウに表示されます。エラーがあった場合は，アウトプットウィンドウに文法エラーと警告が表示されます。

図 2-32 アウトプットウィンドウ



アウトプットウィンドウは，［表示］メニューの［アウトプット］を選択して，表示／非表示を切り替えることができます。

2.7.2 デバッグ

AIT ファイルのデバッグを容易にするために，特定の位置で AIT ファイルの実行を停止できます。AIT

ファイルの実行が停止すると，監視ウィンドウでは変数の値が表示または更新されます。

AIT ファイルは，次の単位で実行できます。

- 設定したブレークポイントまで
- カーソル行の前まで
- ステートメント単位

(1) ブレークポイントを設定する

ブレークポイントとは，デバッグプロセスを停止するポイントです。ブレークポイントは任意の行に追加でき，有効にしたり無効にしたりできます。ブレークポイントが行内に設定されている場合，それが有効な状態であれば，編集ウィンドウの左側にカラーの丸が表示され，無効な状態であれば白丸が表示されます。不要なブレークポイントは削除できます。また，設定したブレークポイントは，AIT ファイルを閉じると解除されます。

ブレークポイントの追加，削除，有効化，および無効化は，［ブレークポイントの設定］または［ブレークポイントの追加／削除］メニューで実行します。

(a) ［ブレークポイントの設定］を使って，ブレークポイントを設定する

［ブレークポイントの設定］ダイアログボックスで，ブレークポイントを追加，削除，有効化，無効化する方法を説明します。

1. AIT **ファイルを開き，［デバッグ］－［ブレークポイントの設定］を選択する。**
   ［ブレークポイントの設定］ダイアログボックスが表示されます。

図 2-33 **［ブレークポイントの設定］ダイアログボックス**

ブレークポイント行番号

ブレークポイントを追加したい行番号を入力し，［追加］ボタンをクリックすると，指定した行にブレークポイントが追加されます。

ブレークポイント

アクティブな AIT ファイルに設定されているすべてのブレークポイントが表示されます。行番号に対応するチェックボックスをオンにすると，ブレークポイントは有効になり，オフにすると無効になります。

［削除］ボタン

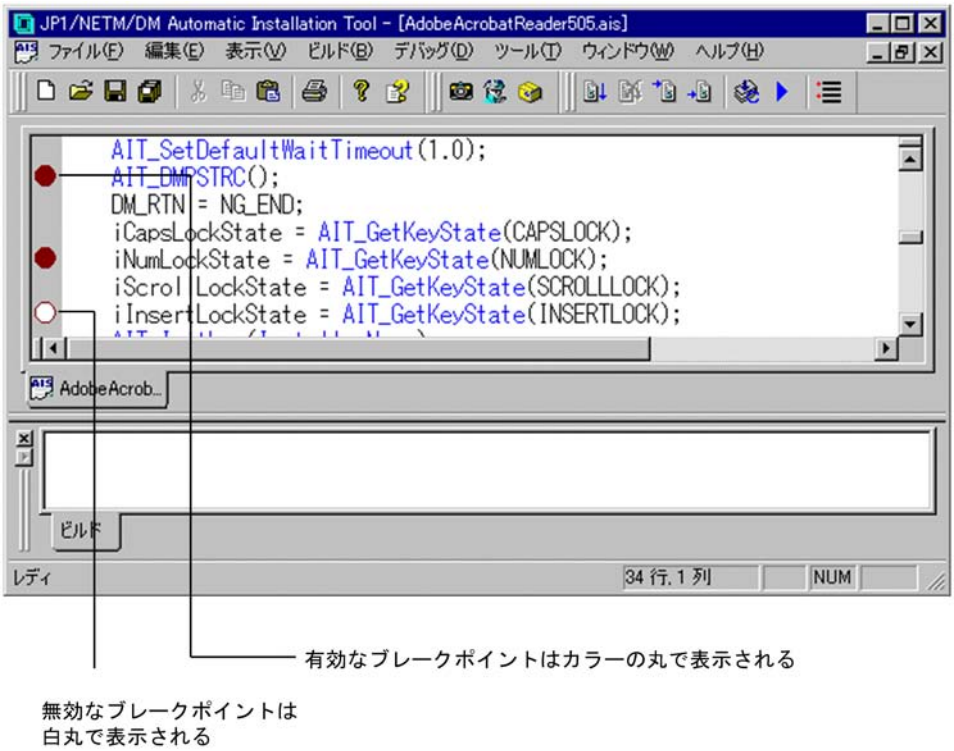
「ブレークポイント」欄で選択した行番号のブレークポイントを削除します。

［すべて削除］ボタン

設定しているすべてのブレークポイントを削除します。

2. **ブレークポイントを設定して，［OK］ボタンをクリックする。**
［ブレークポイントの設定］ダイアログボックスでの設定が，編集ウィンドウに反映されます。編集ウィンドウを次の図に示します。

図 2-34　編集ウィンドウ



(b) ［ブレークポイントの追加／削除］を使って，ブレークポイントを設定する

［ブレークポイントの設定］ダイアログボックスを使用しないで，編集ウィンドウ中で，ブレークポイントを追加，削除，および有効にする方法を次に示します。

1. **AIT ファイルを開き，デバッグプロセスを停止させたい行にカーソルを移動する。**
2. **［デバッグ］－［ブレークポイントの追加／削除］を選択する。**
選択した行によって，次の処理が実施されます。
   - **ブレークポイントのない行**
   ブレークポイントが有効な状態で追加されます。
   - **有効なブレークポイントがある行**
   ブレークポイントが削除されます。
   - **無効なブレークポイントがある行**
   ブレークポイントが有効になります。

   なお，［ブレークポイントの追加／削除］メニューを選択する代わりに，［F9］キーをショートカットキーとして使用することもできます。

(2) 特定の位置まで実行する

現在の位置から特定の位置まで AIT ファイルを実行できます。

特定の位置で実行を停止させるには，次の方法があります。

- ブレークポイントを設定して［デバッグ］－［実行］を選択する。

- 任意の位置にカーソルを移動して［デバッグ］－［カーソル行の前まで実行］を選択する。
- ［デバッグ］－［ステップオーバー］を選択し，ステートメント単位で実行する。

なお，デバッグを終了するには［デバッグ］－［デバッグの中断］を選択します。

(a) ブレークポイントまで実行する

設定したブレークポイントまで AIT ファイルを実行します。コメントやブランクだけなどの実行できない行に設定されているブレークポイントは，デバッグが開始されると，その直後の実行可能な行に移動します。

1. AIT **ファイルの任意の位置にブレークポイントを設定する。**
   ブレークポイントの設定方法については，「(1) ブレークポイントを設定する」を参照してください。
2. **［デバッグ］－［実行］を選択する。**
   最初のブレークポイントまでステートメントが実行され，ブレークポイント行にデバッグカーソルが移動します。
3. **操作 2 を繰り返して次のブレークポイントまで実行するか，そのほかの［デバッグ］メニュー項目を選択する。**

(b) カーソル行の前まで実行する

カーソルのある行の前まで AIT ファイルを実行します。

1. AIT **ファイルの任意の位置にカーソルを移動する。**
2. **［デバッグ］－［カーソル行の前まで実行］を選択する。**
   カーソル行の前までステートメントが実行され，操作 1 で指定した位置にデバッグカーソルが移動します。
3. **操作 1 と操作 2 を繰り返して次のカーソル行の前まで実行するか，そのほかの［デバッグ］メニュー項目を選択する。**

(c) ステートメント単位で実行する

AIT ファイルの終わりまで，または「デバッグの中断」を選択してデバッグプロセスを中断するまで，スクリプトを 1 ステートメントずつ実行できます。

1. AIT **ファイルを開く。**
2. **［デバッグ］－［ステップオーバー］を選択する。**
   最初のステートメント行が実行され，デバッグカーソルは次のステートメント行に移動します。
3. **操作 2 を繰り返して現在のステートメント行を実行するか，そのほかの［デバッグ］メニュー項目を選択する。**

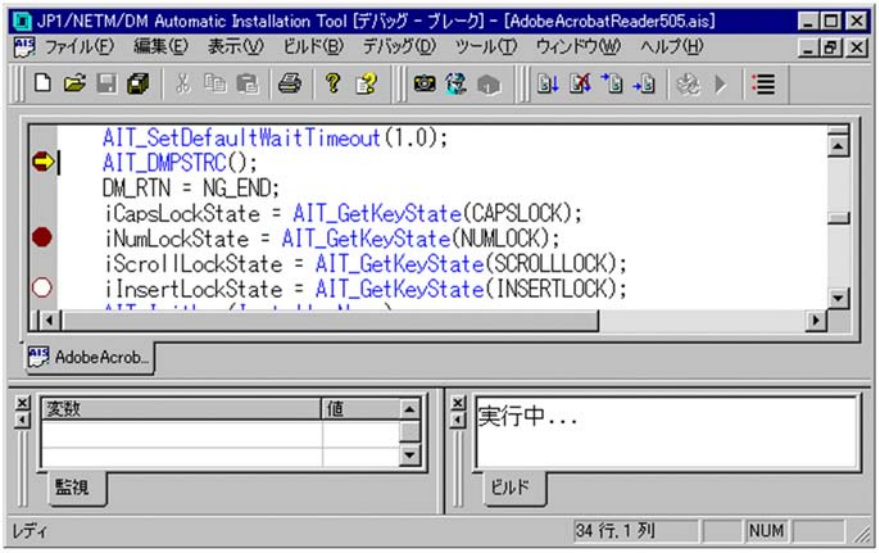
(3) 変数を監視，変更する

AIT ファイルをデバッグするとき，指定した変数の値を監視ウィンドウに表示して，監視できます。特定の位置で AIT ファイルの実行が停止したとき，現在の変数の値が監視ウィンドウに表示されます。監視ウィンドウを使用して，変数の値を変更することもできます。

次に，監視ウィンドウで，指定した変数を監視する方法を示します。

1. AIT **ファイルのデバッグ時に，［表示］－［監視ウィンドウ］を選択する。**
   監視ウィンドウが表示されます。

図 2-35 監視ウィンドウ



2. **「変数」欄に変数名を入力し，[Enter] キーを押す。**
   編集ウィンドウから監視ウィンドウに，変数をドラッグ＆ドロップして，変数名を入力することもできます。
   監視ウィンドウには，変数の値が表示されます。数値データは，デフォルトでは 10 進数で表示されますが，右クリックして表示されるメニューから［16 進数表示］を選択して，16 進数で表示することもできます。文字列データは文字列定数で表示されます。
   監視ウィンドウで指定した変数名が AIT ファイルで定義されていない場合は，エラーメッセージが表示されます。
3. **特定の位置まで，AIT ファイルを実行する。**
   現在の変数の値が監視ウィンドウに表示されます。
4. **変数の値を変更したい場合は，監視ウィンドウで新しい値を入力して，[Enter] キーを押す。**
   AIT ファイル内の変数の値が変更されます。

監視が不要になった変数を監視ウィンドウから削除するには，監視ウィンドウで削除したい変数を選択し，［Delete］キーを押します。

3 AIT 言語リファレンス

この章では，AIT ファイルの形式と，AIT 言語の構文について説明します。

3.1 AIT ファイルの形式

AIT ファイルは，次の四つのセクションで構成されます。

- PACKAGE_INFO
- DEFINE
- MAIN
- ERROR

これらすべてのセクションは必須です。また，セクションの順序は変更できません。

AIT ファイルの形式を次に示します。AIT 言語では，大文字と小文字が区別されません。また，スクリプト中にはコメントを記述できます。

AIT ファイルの形式

```
PACKAGE_INFO
{
   // パッケージ情報
}

DEFINE
{
   // 変数と定数の定義および初期化
}

MAIN
{
   // ウィンドウに対する操作
}

ERROR
{
   // エラー発生時の処理
}
```

3.2 セクション

AIT ファイルに記述するセクションについて説明します。

3.2.1 PACKAGE_INFO

配布するソフトウェアの，パッケージ情報とセットアップに必要な情報を指定します。

このセクションで指定する項目を次に示します。

(1) 形式

次に示す項目を指定できます。これら以外は指定できません。

```
PackageID = "パッケージ識別ID";
Product  =  "パッケージ名";
Version = "バージョン/リビジョン";
InstallerName = "インストールプログラム名";
InstallDrive = "インストール先ドライブ";
InstallDirectory = "インストール先ディレクトリ";
IconGroupName = "アイコングループ";
SerialNumber = "シリアルナンバー";
EUser = "英語の所有者名";
ECompany = "英語の会社名";
JUser = "日本語の所有者名";
JCompany = "日本語の会社名";
ScriptFileVersion = "AITファイルのバージョン";
```

なお，「EUser」と「ECompany」の組み合わせと，「JUser」と「JCompany」の組み合わせは，どちらか一組しか指定できません。

(2) 記述する項目

PACKAGE_INFO セクションに記述する各項目について説明します。どの項目にも，「¥n」，「¥r」，「¥t」のような特殊な意味を持つ文字は使用できません。

項目名	内容	省略
PackageID	パッケージ識別 ID を，1 ～ 44 バイトで指定します。 半角英数字（英字は大文字だけ），「-」（ハイフン），または「_」（アンダーバー）で指定します。	不可
Product	パッケージ名を，1 ～ 50 バイトで指定します。 「¥」は使用できません。	不可
Version	ソフトウェアのバージョン / リビジョンを，1 ～ 6 バイトで指定します。 英数字（英字は大文字だけ）と「/」（スラッシュ）で指定します。	不可
InstallerName	ソフトウェアをインストールするときの，インストールプログラム（インストーラ）の名前を，1 ～ 256 バイトで指定します。 次に示す記号は使用できません。 「*」「"」「:」「|」「<」「>」「?」	不可
InstallDrive	ソフトウェアをインストールするドライブを，2 バイトで指定します。 半角英数字および「:」（コロン）で指定します。	不可
InstallDirectory	ソフトウェアをインストールするディレクトリを指定します。 「¥」で始まるパス名を，1 ～ 128 バイトで指定します。	不可
IconGroupName	ソフトウェアのアイコングループを，1 ～ 40 バイトで指定します。	可

項目名	内容	省略
SerialNumber	インストールするソフトウェアのシリアルナンバーを，1 ～ 64 バイトで指定します。 インストール時に CD キーの必要なソフトウェアは，CD キーを入力してください。	可
EUser	英語版のソフトウェアの場合に，ソフトウェアの所有者名を 1 ～ 40 バイトで指定します。 半角英数字で指定します。	可
ECompany	英語版のソフトウェアの場合に，ソフトウェアを所有する会社名を 1 ～ 80 バイトで指定します。 半角英数字で指定します。	可
JUser	日本語版のソフトウェアの場合に，ソフトウェアの所有者名を 1 ～ 40 バイトで指定します。	可
JCompany	日本語版のソフトウェアの場合に，ソフトウェアを所有する会社名を 1 ～ 80 バイトで指定します。	可
ScriptFileVersion	「*nn.nn.nn.nn*」の形式で，数字で指定します。「.」（ドット）は連続して指定できません。「.」（ドット）で区切った四つの *nn* はすべて記述してください。四つ目の区切りの値が指定されていない場合，その区切りはゼロと見なされます。例えば，"1.0.0." は，"1.0.0.0" として扱われます。 この情報を指定することで，Automatic Installation Tool の新しいバージョンで作成された AIT ファイルが，古いバージョンの実行エンジンで実行されなくなります。 なお，AIT ファイルのバージョンが PACKAGE_INFO セクションに記述されていない場合は，アクティブな実行エンジンの DLL のバージョンが AIT ファイルのバージョンとして取得されます。	可

(3) 記述例

```
PACKAGE_INFO
{
   PackageID = "D";
   Version = "1";
   Product = "パッケージ名";
   InstallerName = "インストールプログラム名";
   InstallDrive = "D:";
   InstallDirectory = "¥Plan14.1";
   JUser = "パッケージの所有者名";
   JCompany = "パッケージを所有する会社名";
   SerialNumber = "パッケージのシリアルナンバー";
   IconGroupName= "アイコングループ名";
   ScriptFileVersion = "1.0.0.0";
}
```

3.2.2 DEFINE

AIT ファイルに記述するすべての変数および定数を定義します。このセクションでは，変数と定数の宣言および初期化だけができます。変数および定数以外の構文は指定できません。

(1) 記述例

```
DEFINE
{
    const integer OK_END = 0, NG_END = -1 ;
    string sMsgText;
    float TimeOut;
    bool sInvalidPathFlag = false;
}
```

(2) 備考

- 変数の再定義はできません。
- 宣言していない変数を使用すると，文法チェック時に，アウトプットウィンドウに警告メッセージが表示されます。

3.2.3 MAIN

ソフトウェアの自動インストールに必要な処理を記述します。このセクションでは，データ型宣言以外のすべての構文を使用できます。

また，インストール結果のリターンコードを JP1/NETM/DM Client に通知する指定もできます。

(1) 記述例

```
MAIN
{
    AIT_SetDefaultWaitTimeout(1.0);
    AIT_DMPSTRC();
    DM_RTN = NG_END;
    AIT_InitLog(InstallerName);
    bRtn= AIT_Exec(InstallerName,SW_SHOWNORMAL);
    if(bRtn == false)
        iErrorNo = AIT_GetLastError();
        strErrorTxt = AIT_GetErrorText(iErrorNo);
        AIT_LogMessage(strErrorTxt);
        iLoopCount = iLoopMax;
    else
        AIT_LogMessage("Recorder File started");
    Endif;
    while(iLoopCount < iLoopMax)
        if(AIT_FocusWindow("check", "#32770") > 0)
            AIT_LogMessage("Window - Caption: check, Class Name: #32770");
            AIT_CtrlSetFocus("チェックボックス(&C)",CHECKBOX_CTRL);
            AIT_Sleep(SLEEP_TIME_EVENTS);
            AIT_LogMessage("AIT_CtrlSetFocus('"チェックボックス
(&C)'",CHECKBOX_CTRL);");
            AIT_PlayKey("{ENTER}");
            AIT_Sleep(SLEEP_TIME_EVENTS);
            AIT_LogMessage("AIT_PlayKey('"{ENTER}'");");
            iLoopCount = iLoopMax;
            DM_RTN = OK_END;
            continue;
        Endif;
        iLoopCount = iLoopCount + 1;
        AIT_Sleep(SLEEP_TIME);
    loop;
    if(DM_RTN == OK_END)
        AIT_LogMessage("Recorder File ended normally");
    else
        AIT_LogMessage("Recorder File ended Abnormally");
    Endif;
    WINH = AIT_RegisterWindowMessage("DMP_REC_QUIT");
    if(WINH != 0)
        AIT_PostMessage(HWND_BROADCAST,WINH,DM_RTN,0);
    Endif;
}
```

3.2.4 ERROR

実行の異常終了時に実行されるステートメントを記述します。インストール結果のリターンコードを JP1/NETM/DM Client に通知する指定ができます。

AIT ファイルの実行時にエラーが発生すると，このセクションに実行制御が移ります。

(1) 記述例

```
ERROR
{
   WINH = AIT_RegisterWindowMessage("DMP_REC_QUIT");
     if  (WINH != 0)
       AIT_PostMessage (HWND_BROADCAST,WINH,-1,0);
     endif;
}
```

3.3 データ型

AIT ファイルに記述するデータ型について説明します。以降に示す基本データ型がサポートされています。

3.3.1 integer 型

integer 型は，-2,147,483,648 ～ +2,147,483,647 の数字を含む基本データ型です。数字に小数や指数を含むことはできません。

キーワード「integer」を使用すると，integer 型変数の宣言や，変数や定数を初期化できます。このキーワードは DEFINE セクションだけで使用できます。複数の変数を宣言するときは，変数を「,」（コンマ）で区切ってください。

また，キーワード「integer」と組み合わせてキーワード「const」を使用すると，定数を宣言できます。DEFINE セクションで宣言した定数は，値を参照できるようになりますが，MAIN セクションや ERROR セクションで値を変更することはできません。

(1) 形式

```
DEFINE
{
   [const] integer variable_name1 [= integer_constant1] [, variable_name2 [=
integer_constant2] ] ;
}
```

(2) 記述例

```
DEFINE
{
    const integer OK_END = 0, NG_END = -1 ; // 有効
    integer end_status, return_code;        // 0に初期化
}
```

(3) 備考

- 変数や定数を初期化する場合，初期化の値は 10 進数でだけ指定できます。
- integer 型変数に割り当てられる値は，integer 型，bool 型，float 型のどれかです。ただし，値が float 型の場合は正確性を欠くおそれがあるため，文法チェック時に，アウトプットウィンドウに警告メッセージが表示されます。
- すべての変数は，デフォルトで 0 に初期化されます。(2) の例での「end_status」および「return_code」は，0 で初期化されています。

3.3.2 float 型

float 型は，32 ビット浮動小数点数を含む基本データ型です。float 型変数および定数は，+3.40282347e+38 ～ +1.175494351e-38 の絶対値を持てます。

キーワード「float」を使用すると，float 型変数または定数を宣言し初期化できます。このキーワードは DEFINE セクションでだけ使用できます。複数の変数を宣言するときは，変数を「,」（コンマ）で区切ってください。

また，キーワード「float」と組み合わせて，キーワード「const」を使用すると，定数を宣言できます。DEFINE セクションで宣言した定数は，値を参照できるようになりますが，MAIN セクションや ERROR セクションで値を変更することはできません。

(1) 形式

```
DEFINE
{
    [const] float variable_name1 [= float_constant1] [, variable_name2 [=
float_constant2] ] ;
}
```

浮動小数点数には，小数点または指数表現（"E" または "e"）を含められます。指数部には，"E" または "e" と，その後に整数値を指定します。整数値には符号（"+" または "-"）を付けられます。浮動小数点定数には，1 けた以上の数字が必要で，小数点または指数を指定する必要があります。

(2) 記述例

```
DEFINE
{
    float DefaultTimeOut = 0.01, DefaultSleep=5.0 ; // 有効
    float SleepMax;                                 // 有効
}
```

(3) 備考

- 浮動小数点定数は，10 進数でだけ指定できます。
- float 型変数に割り当てられる値は，bool 型，float 型，integer 型のどれかです。
- float 型変数は，デフォルトで 0 に初期化されます。(2) の例での「SleepMax」は，0 で初期化されています。
- float 型で有効なけた数は 11 です（小数点を含む）。

3.3.3 bool 型

bool 型は，true（真）および false（偽）という値を持つデータ型です。true と false の間には，次の関係が成り立ちます。

- 「!false」は「true」と同じ意味
- 「!true」は「false」と同じ意味

キーワード「bool」を使用すると，bool 型変数や定数を宣言し初期化できます。このキーワードは，DEFINE セクションでだけ使用できます。複数の変数を宣言するときは，これらの変数を「,」（コンマ）で区切ってください。

また，キーワード「bool」と組み合わせて，キーワード「const」を使用すると，定数を宣言できます。DEFINE セクションで宣言した定数は，値を参照できるようになりますが，MAIN セクションや ERROR セクションで値を変更することはできません。

なお，比較によって 0 と評価された式は false，0 以外の数値と評価された式は true として解釈されます。

(1) 形式

```
DEFINE
{
    [const] bool variable_name1 [= true|false] [, variable_name2 [= true|false]
] ;
}
```

(2) 記述例

```
DEFINE
{
    bool sInvalidPathFlag, sDirectorySetFlag = true;
    bool SMemoryInsuff = false;
    bool sEndGUI = false;
```

```
}
```

(3) 備考

- bool 型変数に割り当てられる値は，integer 型，float 型のどちらかです。ただし，どちらの場合も正確性を欠くおそれがあるため，文法チェック時に，アウトプットウィンドウに警告メッセージが発行されます。
- すべての変数は，デフォルトで false に初期化されます。(2) の例での「sInvalidPathFlag」は，false で初期化されています。

3.3.4 string 型

string 型は，可変長の文字列を持つ基本データ型です。

キーワード「string」を使用すると，string 型変数や定数を宣言し初期化できます。このキーワードは，DEFINE セクションでだけ使用できます。複数の変数を宣言するときは，変数を「,」（コンマ）で区切ってください。

また，キーワード「string」と組み合わせて，キーワード「const」を使用すると，定数を宣言できます。DEFINE セクションで宣言した定数は，値を参照できるようになりますが，MAIN セクションや ERROR セクションで値を変更することはできません。

文字列定数を指定するには，文字を「"」（ダブルクォーテーション）で囲みます。文字列内部で「"」を使用するときは，直前に「'」（シングルクォーテーション）を指定してください。

(1) 形式

```
DEFINE
{
    [const] string variable_name1 [= "StringValue"] [, variable_name2 [=
integer_constant2] ] ;
}
```

(2) 記述例

```
DEFINE
{
    string CaptionName="Setup";  // 有効
    string ErrorText;            // 有効
}
```

(3) 備考

- string 型変数は，連結操作の結果として文字列長が増えることがあります。
- すべての変数は，デフォルトで空になります。(2) の例での「ErrorText」は空になります。
- string 型は，行末に「_」（アンダーバー）を記述することで次の行に続けることができます。次に示す例のように記述した場合，文字列変数「ErrorText」の値は「sample Testing success」になります。

  例

```
DEFINE
{
   string ErrorText = "sample_
 Testing_
 success";   // stringは複数行に続いています。
}
```

- 次に示す文字は，文字列内で特殊な文字として扱われます。

文字	扱い
¥n	改行
¥r	復帰
¥t	タブ文字
'¥	「¥」
'"	「"」
''	「'」

文字列変数「ErrorText」に「"Sample Testing"」を代入する場合の記述例を次に示します。

例

```
DEFINE
{
   string ErrorText = "'"Sample Testing'"";
          // 「Sample Testing」を「'"」で囲みます。
}
```

文字列変数「Path」に「C:¥Windows¥system32」を代入する場合の記述例を次に示します。

例

```
DEFINE
{
   string Path = "C:'¥Windows'¥system32";
          // 「C:¥Windows¥system32」を「Path」に代入します。
}
```

3.4 演算子

演算子は，次の評価をするために指定します。

● 一つのオペランド（単項演算子）

● 二つのオペランド（二項演算子）

演算子を含む複数の式が評価される順番は，厳密な優先順位に従って定義されています。演算子は，左のオペランドと右のオペランドのどちらかと結合します。これを「結合順序」と呼びます。同じグループ内の演算子は同じ優先順位を持ち，「()」（丸括弧）を使用して明示的に優先順位を変更しないかぎり，オペランドの左から右に評価されます。

AIT 言語でサポートしている演算子を表 3-1 および表 3-2 に示します。

表 3-1 AIT 言語でサポートしている単項演算子

単項演算子	意味
+	単項プラス
-	単項マイナス
!	単項否定

表 3-2 AIT 言語でサポートしている二項演算子

二項演算子	意味
+ -	加法演算子
* / %	乗除演算子
< <= > >= != ==	比較演算子
& |	ビット単位演算子
&& ||	論理演算子

3.4.1 代入操作

代入操作では，右オペランドの値を左オペランドに割り当てます。代入操作の左オペランドは定数ではありません。

(1) 形式

代入ステートメント

```
assignment_expression END_STMT
```

代入式

```
identifier assign_operand expression
```

(2) 説明

代入操作では，右の値の型は左の値の型に変換され，割り当て後は左オペランドに値が格納されます。左オペランドは関数または定数にしないでください。

右オペランドと左オペランドの型が異なる場合は，型変更が実行されます。型変更は，指定された演算子と，オペランドまたは演算子の型に応じて実行されます。

AIT 言語で実行される型変更を次の表に示します。警告やエラーは文法チェック時に表示されます。

左オペランドの型	右オペランドの型	結果	説明
integer	integer	integer	integer 型変数の値が別の integer 型変数に代入されます。
integer	float	integer	切り捨て後の値が integer 型変数に代入されます。データの正確性が損なわれるため警告が表示されます。
integer	bool	integer	true は 1 として，false は 0 として integer 型変数に代入されます。
integer	string	エラー	string 値は integer に型変更できないため，エラーが表示されます。
float	integer	float	integer 型変数の値が float 型変数に代入されます。
float	float	float	float 型変数の値が別の float 型変数に代入されます。
float	bool	float	true は 1.0 として，false は 0.0 として float 型変数に代入されます。
float	string	エラー	string 値は float に型変更できないため，エラーが表示されます。
bool	integer	bool	0 以外の値は true として，0 は false として bool 型変数に代入されます。データが失われるため警告が表示されます。
bool	float	bool	
bool	bool	bool	bool 型変数の値が別の bool 型変数に代入されます。
bool	string	エラー	string 型変数の値は bool に型変更できないため，エラーが表示されます。
string	integer	エラー	integer 型変数の値は string に型変更できないため，エラーが表示されます。
string	float	エラー	float 型変数の値は string に型変更できないため，エラーが表示されます。
string	bool	エラー	bool 型変数の値は string に型変更できないため，エラーが表示されます。
string	string	string	string 型変数の値が別の string 型変数に代入されます。

なお，AIT 言語では多重代入ができます。多重代入によって，ある値を二つの変数に代入できます。

(3) 記述例

```
MAIN
    {
        a = 10;          // 10という値がaに代入されます。
        a = b = 20;      // 多重代入が指定されています。
                         // 20という値が始めに変数bに代入され，
                         // 次に変数aに代入されます。
    }
```

3.4.2 単項プラス

単項プラスは，数式の正の値を戻す単項演算子です。

(1) 形式

```
+[(]式[)]
```

(2) 説明

単項プラス演算子（+）のオペランドは，算術型でなければなりません。単項演算子はオペランドの前に配置され，右から左に結合します。

正の単項演算子を負の値に対して指定すると，戻り値は負の値になります。負の単項演算子を負の値に対して指定すると，戻り値は正の値になります。

(3) 記述例

```
const integer NG_END = +1;          // DEFINEセクション内のステートメント
sloopcnt = +(sloopmin - sloopmax); // MAINセクション内のステートメント
```

3.4.3 単項マイナス

単項マイナスは，数式の負の値を戻す単項演算子です。

(1) 形式

```
-[(]式[)]
```

(2) 説明

単項マイナス演算子（-）のオペランドは，算術型でなければなりません。単項演算子はオペランドの前に配置され，右から左に結合します。

負の単項演算子を負の値に対して指定すると，戻り値は正の値になります。正の単項演算子を負の値に対して指定すると，戻り値は負の値になります。

(3) 記述例

```
const integer NG_END = -1;            //DEFINEセクション内のステートメント
integer sloopcnt, sloopmax, sloopmin;
sloopcnt = -(sloopmax - sloopmin);    //MAINセクション内のステートメント
```

3.4.4 単項否定

単項否定は，式で論理否定をする単項演算子です。

(1) 形式

```
!(式)
```

(2) 説明

オペランドの否定が true（例えば，オペランドが false）の場合は true を戻します。逆に，オペランドの否定が false（例えば，オペランドが true）の場合は false を戻します。

単項演算子はオペランドの前に配置され，右から左に結合します。

(3) 記述例

```
bool IsLastDialog;
IsLastDialog = false;
if (!IsLastDialog)
   AIT_LogMessage("Installable Software Extracting... は開かれています");
else
   AIT_LogMessage("Installable Software Extracting... は開かれていません");
endif;
```

3.4.5 加法演算子

加法演算子は，加算（+）および減算（-）を実行します。

(1) 形式

加算式

```
式 + 式
```

除算式

```
式 - 式
```

(2) 説明

加法演算子は，integer 型オペランドや float 型オペランドの通常の算術変換をします。変換結果の型はオペランドの型になります。

- 加算演算子（+）は，二つのオペランドを加算します。両方のオペランドが string 型のときは，二つの文字列が連結されます。片方のオペランドだけが string 型のときは，エラーになります。
- 減算演算子（-）は，一つ目のオペランドから二つ目のオペランドを減算します。両方のオペランドが数値である必要があります。片方または両方のオペランドが string 型の場合は，結果がエラーとなります。

(3) 記述例

```
MAIN
{
        ...
        sloop_cnt = sloop_cnt+1;
        sloop_cnt = sloop_cnt-1;
        ...
        ...
}
```

(4) 備考

加法演算子がする変換処理は，オーバーフローやアンダーフロー状態に対応できないため，加法演算子の変換結果がオペランドの型で示されない場合は，情報が欠落するおそれがあります。

3.4.6 乗除演算子

乗除演算子は，乗算（*），除算（/），および剰余算（%）を実行します。

(1) 形式

乗算式

式 * 式

除算式

式 / 式

剰余算式

式 % 式

(2) 説明

乗除演算子は，オペランドの通常の算術変換をします。変換結果のデータ型は，オペランドのデータ型になります。両方のオペランドが数値である必要があります。オペランドの片方または両方が string 型の場合は，結果がエラーとなります。

- 乗算演算子（*）は，二つのオペランドを乗算します。オペランドは integer 型でも float 型でもよく，型が異なっていてもかまいません。
- 除算演算子（/）は，一つ目のオペランドを二つ目のオペランドで除算します。オペランドは integer 型でも float 型でもよく，型が異なっていてもかまいません。0 で除算した場合の結果は未定義となりますが，文法チェック時またはランタイム時にエラーが発行されます。また，両方のオペランドが正の値または符号なしのときは，結果は 0 に切り捨てられます。
- 剰余演算子（%）オペランドは，整数でなければなりません。評価結果は，一つ目のオペランドが二つ目のオペランドで除算したあとの剰余になりますが，割り切れなければ評価結果は次の規則によって決まります。
  - 右オペランドが 0 の場合は，結果は未定義となります。
  - 両方のオペランドが正の値または符号なしの場合は，結果は正の値となります。

(3) 記述例

```
MAIN
{
if (sloop_cnt > 10)
   AIT_Sleep(SLEEP_TIME / 2);
else
   AIT_Sleep(SLEEP_TIME * 2);
endif;
}
```

(4) 備考

- 乗除演算子による変換処理は，オーバーフローやアンダーフロー状態に対応できないため，乗除演算子の変換結果がオペランドの型で示されない場合は，情報が欠落するおそれがあります。
- ランタイム時に 0 で除算すると，プログラムは ERROR セクションに制御を移します。

3.4.7 比較演算子

二項比較演算子は，第 1 オペランドを第 2 オペランドと比較し，指定された関係の妥当性を検証します。比較式の結果は，結果の評価が true の場合は 1 となり，false の場合は 0 となります。結果の型は bool です。

(1) 形式

比較式

```
式 < 式
式 > 式
式 <= 式
式 >= 式
```

等式

```
式 == 式
```

不等式

```
式 != 式
```

(2) 説明

比較演算子による検査の対象となる関係を次に示します。

演算子	検査の対象となる関係
<	一つ目のオペランドが二つ目のオペランドより小さい
>	一つ目のオペランドが二つ目のオペランドより大きい
<=	一つ目のオペランドが二つ目のオペランドより小さいか等しい
>=	一つ目のオペランドが二つ目のオペランドより大きいか等しい
==	一つ目のオペランドが二つ目のオペランドと等しい
!=	一つ目のオペランドが二つ目のオペランドと等しくない

オペランドは，integer 型，float 型，または string 型で指定できます。オペランドの型が異なってもかまいません。比較演算子は，integer 型および float 型オペランドの通常の算術変換をします。さらに，オペランドの型と比較演算子や等号演算子を組み合わせて使うことができます。

次の表では，比較演算子と，比較結果が true，false，または null であるかどうかを判別するための状態を示します。

演算子	意味	true の場合	false の場合
<	より小さい	式 1 < 式 2	式 1 >= 式 2
>	より大きい	式 1 > 式 2	式 1 <= 式 2
<=	より小さいか等しい	式 1 <= 式 2	式 1 > 式 2
>=	より大きいか等しい	式 1 >= 式 2	式 1 < 式 2
==	等しい	式 1 == 式 2	式 1 != 式 2
!=	等しくない	式 1 != 式 2	式 1 == 式 2

次の表では，式の型に応じた式の比較結果を示します。

式の型	比較結果
式の両方が数値の場合	数値を比較する
式の両方が string 型の場合	文字列を比較する
一方の式が数値でもう一方が string 型の場合	エラーとなる

(3) 記述例

```
if (sloop_cnt < (sloop_max - 25))  // <
    AIT_LogMessage("Searching for Active windows");  //Search Active windows
   if (AIT_FocusWindow("Installable Software Extracting...", "#32770",0.0) > 0)
// >
        AIT_LogMessage("Installable Software Extracting... is opened");
        sloop_cnt= 0;
    endif;
endif;
```

3.4.8 ビット単位演算子

ビット単位演算子は，ビット単位 AND（&）演算，およびビット単位 OR（|）演算を実行します。

(1) 形式

ビット単位 AND 演算式

式 & 式

ビット単位 OR 演算式

式 | 式

(2) 説明

ビット単位演算子のオペランドは，integer 型を必ず持たなければなりませんが，型が異なってもかまいません。ビット単位演算子は，通常の算術変換をします。変換結果の型は，オペランドの型になります。

ビット単位演算子の説明を次に示します。

演算子	説明
&	ビット単位 AND 演算子は，一つ目のオペランドの各ビットを二つ目のオペランドの対応するビットと比較します。両方のビットが 1 であれば，比較後の結果ビットを 1 に設定します。これ以外の場合は，比較後の結果ビットを 0 に設定します。
|	ビット単位 OR 演算子は，一つ目のオペランドの各ビットを二つ目のオペランドの対応するビットと比較します。どちらか一方のビットが 1 の場合は，比較後の結果ビットを 1 に設定します。これ以外の場合は，比較後の結果ビットを 0 に設定します。

● ビット単位 AND 演算子は，二つの数式の同じ位置にあるビットをビット単位で比較して，次の表に基づき比較後の結果ビットを設定します。

式 1 のビット	式 2 のビット	比較結果
0	0	0
0	1	0
1	0	0
1	1	1

● ビット単位 OR 演算子は，二つの数式の同じ位置にあるビットをビット単位で比較して，次の表に基づき比較後の結果ビットを設定します。

式 1 のビット	式 2 のビット	比較結果
0	0	0
0	1	1
1	0	1
1	1	1

3.4.9 論理演算子

論理演算子は，論理 AND 演算（&&），および論理 OR 演算（||）を実行します。

(1) 形式

論理 AND 演算式

```
式1 && 式2
```

論理 OR 演算式

```
式1 || 式2
```

(2) 説明

論理演算子は通常の演算変換は実行しません。その代わりに，0 と一致するかどうかという観点で各オペランドを評価します。論理演算の結果は true または false です。結果の型は bool です。

論理演算子の説明を次に示します。

演算子	説明
&&	両方のオペランドが true の場合，結果は true になります。どちらか一方のオペランドが false の場合，結果は false になります。論理 AND 演算の初めのオペランドが false の場合，二つ目のオペランドは評価されません。
||	両方のオペランドの評価が false の場合，結果は false になります。どちらか一方のオペランドの評価が true の場合，結果は true になります。論理 OR 演算の初めのオペランドが true の場合，二つ目のオペランドは評価されません。

論理 AND および論理 OR 式のオペランドは，左から右に評価されます。一つ目のオペランドの値だけで演算結果を判別できるときは，二つ目のオペランドは評価されません。これを「短絡評価」と呼びます。

(a) 論理 AND 演算子（&&）の場合

両方の式が true と評価された場合は，結果は true になります。式のどちらかが false と評価された場合

は，結果は false になります。次の表では，評価結果の判別方法を示します。

式 1 の評価	式 2 の評価	評価結果
true	true	true
true	false	false
false	true	false
false	false	false

(b) 論理 OR 演算子（||）の場合

どちらか一方，または両方の式が true と評価された場合は，結果は true になります。次の表では，評価結果の判別方法を示します。

式 1 の評価	式 2 の評価	評価結果
true	true	true
true	false	true
false	true	true
false	false	false

(3) 記述例

```
DEFINE
{
   float varfloat1 = 1.567e-1;
   integer varint1 = 10;
   integer varfloat2 = 0;
   integer varint2 = 0;
   bool varbool;
   integer WINH;
}
MAIN
{
   varbool = varfloat1 && varint1;
   AIT_LogMessage("The expected value of varbool is: true");
   varbool =varfloat2 && varint1;
   AIT_LogMessage("The expected value of varbool is: false");
   varbool=varfloat1 && varint2;
   AIT_LogMessage("The expected value of varbool is: false");
   varbool=varfloat2 && varint2;
   AIT_LogMessage("The expected value of varbool is: false");
}
```

3.4.10 演算子の優先順位

演算子の優先順位および結合順序は，式のオペランドのグループ化や評価するうえで影響を与えます。演算子の優先順位は，同じ式の中に演算子の優先順位が高い，または低い別の演算子が含まれている場合にだけ意味を持ちます。式の中に高い優先順位の演算子が含まれているときは，その演算子が先に評価されます。演算子の優先順位が等しいときは，評価される順位はそれぞれが持つ結合順序によって決まります。

次の表では，演算子の優先順位と結合順序を示します。演算子は，優先順位の降順で表しています。

記号	演算子の型	結合順序
( )	括弧	左から右
+ - !	単項	右から左
* / %	乗除	左から右

記号	演算子の型	結合順序
+ -	加法	左から右
< > <= >=	比較	左から右
== !=	等号	左から右
&	ビット単位 AND	左から右
|	ビット単位 OR	左から右
&&	論理 AND	左から右
||	論理 OR	左から右
=	代入	左から右

3.5 変数と定数

変数と定数は，どちらもデータを象徴するものです。変数は値が変化するのに対し，定数は値を変更できません。

変数および定数は，宣言することで実体を持ち，使用できるようになります。これらの宣言は必ず AIT ファイルの DEFINE セクションでします。宣言は一度しかできません。変数名および定数名は，64 文字以内で，必ず英字（A ～ Z）から始まるようにします。「_」（アンダーバー）以外の記号は使えません。また，英字の大文字と小文字は区別されません。

変数名および定数名は，目的や用途を示す，わかりやすいものにすることをお勧めします。キーワード，ラベル名，およびマクロ名は，変数名にしないでください。

(1) 変数の例

```
integer LoopCount;
string CountryName;
bool answer;
float r_nTimeOut;
integer ABC;
integer abc;      //変数を再び定義しているため無効
```

上記の例の「integer」という記述は，リストされている変数が integer 型であることを意味します。つまり，「LoopCount」は整数です。同様に，「CountryName」は文字列です。

(2) 定数の例

```
const integer OK_END = 0;        //DM_RTN:OK_END
const integer NG_END = -1;       //DM_RTN:NG_END
const integer SET_SLEEP_TIME = 2;
```

3.6 プログラムフローの制御

一般に複数のステートメントは，順次に実行されます。ただし，フローを別のステートメントに移すには，以降に示すステートメントを使用します。

3.6.1 goto

goto ステートメントを使用すると，同じセクション内で宣言されている有効なラベルの位置に無条件にジャンプできます。

(1) 形式

ステートメント

ジャンプステートメント
ラベルステートメント

ジャンプステートメント

```
goto 識別子;
```

ラベルステートメント

```
識別子:
```

(2) 説明

goto ステートメントは，深くネストされたループの内側から直接ループを抜けられます。一方，break ステートメントは，繰り返しステートメントの一つのレベルからしか終了できません。

(3) 記述例

```
//MAINセクション
MAIN
{
   ...
   ...
   AIT_LogMessage("LBL030: JUMP TO LABEL");
   goto label1;
       AIT_LogMessage("LBL030: NOT DISPLAYED 1");   // 実行されない

       label1:                      // 制御はここに移る
          AIT_LogMessage("LBL030: JUMPING TO LABEL ");   // 実行される
}
```

(4) 備考

プログラミングスタイルとしては，できるだけ goto ステートメントより break，continue，および return ステートメントを使う方がよいでしょう。

3.6.2 ラベル

goto ステートメントを使用してプログラムの制御を特定のステートメントにダイレクトに移すには，移動先のステートメントにラベルを付ける必要があります。

ラベルを宣言するには，識別子の末尾に「:」（コロン）を付けます。ラベルの宣言は，MAIN セクションまたは ERROR セクションのどの位置でもかまいません。ラベル名は，変数または定数と同じ規則に従い，必ずユニークなものを指定します。また，変数や定数をラベルとして再使用してはいけません。

ラベルは，MAIN セクションまたは ERROR セクション以外では意味がなく，goto ステートメントと関連づけられている場合にだけ意味を持ちます。関連づけられていない場合はラベルが解釈されないでステー

トメントが実行されます。ラベルの付いたステートメントが goto ステートメントと関連づけられていない場合は，警告メッセージが表示されます。

(1) 形式

ラベルステートメント

識別子:

(2) 記述例

```
AIT_MessageBox("ss","xx");
goto label18;
   AIT_LogMessage("LBL040: NOT DISPLAYED 29");    // 実行されない
   label18:          // 制御がここに移る
   AIT_LogMessage("LBL040: INSIDE LABELLED STATEMENT"); // 実行される
   AIT_Exit();
```

3.6.3 if-else-endif

if ステートメントを使用すると，条件分岐を処理できるようになります。

条件が 0 以外の場合は，if ステートメントの本体が実行されます。条件が 0 の場合は，本体以外の部分が実行されます。

このステートメントの本体以外の部分はオプションです。条件は括弧で囲んでください。

(1) 形式

```
if (条件)
    [式1;]
[else
    [式2;]]
endif;
```

(2) 説明

評価の対象は条件です。条件が 0 以外の場合は (1) の式 1 に当たる部分が実行され，条件が 0 の場合は式 2 の部分が実行されます。

if-else-endif ステートメントの else 文節は，対応する else ステートメントが存在しない直前の if ステートメントと関連づけられています。

(3) 記述例

```
if(AIT_FocusWindow("Installable Software-Setup","#32770"))
                          // 次の2行はtrue（0以外）の場合に実行される
   AIT_LogMessage("INSIDE Installable Software SETUP");
   AIT_PlayKey("ENTER");
else
                          // 次の2行はfalse（0）の場合に実行される
   AIT_LogMessage("PROBLEM IN SETUP");
   AIT_Exit();
endif;
```

3.6.4 while-loop

while-loop ステートメントは，指定した条件が false になるまで式を繰り返し実行します。条件は括弧で囲んで指定してください。

(1) 形式

```
while (条件)
    式1;
```

```
    式2;
loop;
```

(2) 説明

1. 条件が評価されます。
2. 条件が最初に false となると，while ステートメントの本体は一度も実行されず，制御が while ステートメントから同じプログラム内の次のステートメントに移ります。
3. 条件が true（0 以外）のときは，ステートメントの本体は実行され，式 1 から実行が繰り返されます。
4. ステートメント本体で break ステートメントが指定されていると，その時点でループは終了します。
5. ステートメント本体で continue ステートメントが指定されていると，それ以降の実行はスキップされ，条件が評価されます。条件が true のときは，実行が繰り返されます。

なお，ネストする while ループの数は，255 以内に抑えてください。

(3) 記述例

```
DEFINE
{
integer WINH,count,length;
float SLEEP_TIME=0.5;
string s1,s2;
integer i,sloop_cnt   = 0;
integer sloop_max = 30;
}
...
...
while ( sloop_cnt < sloop_max)
   AIT_LogMessage("アクティブウィンドウの検索");
   if (AIT_FocusWindow("アンパック中...", "#32770",0.0) > 0)
  if(AIT_FocusWindow("インストールするソフトウェアをアンパック中...", "#32770", 0.0) >
0)
          AIT_LogMessage("インストールするソフトウェアをアンパック中... が開かれました");
      sloop_cnt= 0;
      AIT_Sleep(SLEEP_TIME);
   endif;
   endif;
   AIT_Sleep(SLEEP_TIME);
   sloop_cnt = sloop_cnt + 1;
loop;
```

3.6.5 do-while

do-while ステートメントは，指定された終了条件が false と評価されるまで式を繰り返し実行します。条件は括弧で指定してください。

(1) 形式

```
do
   式1
   式2
   ...
   ...
while (条件);
```

(2) 説明

1. do-while ステートメントの本体が実行されます。
2. 次に，条件が評価されます。条件が false のときは，do-while ステートメントは終了し，制御が同じプログラム内の次のステートメントに移ります。条件が true（0 以外）のときは，式 1 から実行が繰り返されます。
3. ステートメント本体で break ステートメントが指定されていると，その時点でループは終了します。
4. ステートメント本体で continue ステートメントが指定されていると，それ以降の実行はスキップされ，

条件が評価されます。条件が true のときは，実行が繰り返されます。

5. この結果，ループの本体は，最低 1 回は必ず実行されます。

なお，ループ内の break または continue ステートメントの数は，255 以内に抑えてください。また，ネストする do-while ステートメントの数は，255 以内に抑えてください。

(3) 記述例

```
DEFINE
{
integer WINH,count,length;
float SLEEP_TIME=0.5;
string s1,s2;
integer i,sloop_cnt   = 0;
integer sloop_max = 30;
}
...
...
do
   AIT_LogMessage("Searching for Active windows");
   if (AIT_FocusWindow("Installable Software", "#32770",0.0) > 0)
       if(AIT_FocusWindow("Unpacking Installable Software...", "#32770", 0.0) >
0)
            AIT_LogMessage("Unpacking Installable Software... is opened");
            sloop_cnt= 0;
            AIT_Sleep(SLEEP_TIME);
        endif;
   endif;
   AIT_Sleep(SLEEP_TIME);
   sloop_cnt = sloop_cnt + 1;
while ( sloop_cnt < sloop_max);
...
...
```

3.6.6 for-next

for-next ステートメントは，条件が false になるまで式を繰り返し実行します。for-next ステートメントのオプション式を使用すると，for-next ステートメントの実行中に初期設定や値を変更できます。

一般にループが繰り返される回数は，カウンタによって決まります。

(1) 形式

```
for ( [初期化式] ; [条件式] ; [ループ式] )
     式1;
     式2;
next;
```

(2) 説明

1. 初期化式が指定されていると，この式が評価されます。この式は，ループの初期化を指定します。初期化式には，型の制限はありません。
2. 条件式が指定されていると，この式が評価されます。この評価は，それぞれの繰り返しをする前に実行されますが，次の三つの結果が考えられます。
   - 条件式が true（0 以外）の場合は，ステートメントが実行されます。ループ式が指定されていると，次にこの式が評価されます。そして，条件式の評価から実行が繰り返されます。
   - 条件式の指定が省略されているときは，条件式は true と解釈され，上記と同様に実行されます。引数で条件式を指定しないと，ステートメント本体の内部で break ステートメントが実行されたとき，または goto ステートメント（ステートメント本体の外側のラベル付きステートメントに対応したもの）が実行された時に，for ステートメントは終了します。
   - 条件式が false（0）のときは，for-next ステートメントの実行が終了し，制御が同じプログラム内の次のステートメントに移ります。

3. ステートメント本体で break ステートメントが指定されていると，その時点でループは停止します。
4. ステートメント本体で continue ステートメントが指定されていると，それ以降の実行をスキップし，条件式が評価されます。条件が true のときは，実行が繰り返されます。

なお，ネストする for-next 構造体の数は，255 以内に抑えてください。

(3) 記述例

```
DEFINE
{
integer WINH,count,length;
float SLEEP_TIME=0.5;
string s1,s2;
integer i,sloop_cnt   = 0;
integer sloop_max = 30;
}
...
...
sloop_cnt=1;
AIT_LogMessage("Searching for Active windows - For");
for(; sloop_cnt < sloop_max ;sloop_cnt = sloop_cnt + 1)
    if (AIT_FocusWindow("Unpacking", "#32770",0.0) > 0)
       if(AIT_FocusWindow("Unpacking Installable Software...", "#32770", 0.0) >
0)
               AIT_LogMessage("Unpacking Installable Software... is opened");
               sloop_cnt= 0;
               AIT_Sleep(SLEEP_TIME);
        endif;
        endif;
   AIT_Sleep(SLEEP_TIME);
next;
```

3.6.7 continue

continue ステートメントは，これを囲むいちばん小さい do，for，または while ステートメントの次の繰り返し位置に制御を移します。

一般に continue ステートメントは，深くネストされたループの内側からループの開始位置に戻すために使用します。

(1) 形式

ジャンプステートメント

```
continue;
```

(2) 説明

continue ステートメントに達すると，do，for または while ステートメントの次の繰り返し位置は，次のように決定されます。

- do または while ステートメントの内側で，do または while ステートメントの式が再評価されることで，次の繰り返しが開始されます。
- for ステートメントで continue ステートメントが指定されていると，for ステートメントの条件式を再評価して，その結果に応じてステートメント本体の実行を終了するか，または繰り返します。

なお，ループ構造体に含める continue ステートメントの数は，255 以内に抑えてください。

(3) 記述例

```
for (count=1;count<=10;count=count+1)
    if (count < 5)
           continue;
    endif;
```

```
    ...
next;
```

上の例では，continue ステートメント以降のコードは，count が 5 になるまで処理が省略されます。

3.6.8 break

break ステートメントは，これを囲むいちばん近い do-while，for-next，switch-endswitch，または while-loop ステートメントの実行を終了します。制御は，終了したステートメントの後ろにあるステートメントに移ります。break ステートメントは，繰り返しから脱出するために使用します。

(1) 形式

ジャンプステートメント

```
   break;
```

(2) 説明

break ステートメントは，ループ内では，終了基準が評価される前にループを終了するために使用します。また，switch ステートメント内部で，特定の場合に実行を終了するときにも使用します。繰り返しステートメントまたは switch ステートメントを指定しなければ，エラーが発生します。

なお，ループ構造体の中に含める break ステートメントは，255 以内に抑えてください。

(3) 記述例

```
i = 0;
for(;;)
   AIT_LogMessage("Inside Loop");
   i = i + 1;
   if(i>100)
      break;
   endif;
next;
```

3.6.9 switch-endswitch

switch ステートメントを使用すると，式の値に応じてコードにある複数の選択肢の中から処理を選択できます。式は括弧で囲んで指定してください。

switch ステートメントには，case ラベルまたは default ラベルが付いているものがあり，それらが選択肢となります。各 case ラベルは，定数値を持てます。switch ステートメントは case ラベルがなければ意味を持たないため，switch ステートメント中には，一つ以上の case ラベルを付けるようにしてください。

default ラベルは，一つしか指定できません。default ラベルはオプションであり，これに値を指定しないようにしてください。

(1) 形式

```
switch (式)

  [case 定数値:]+
       [式;]*
  ...
  ...
 [default:]
       [式;]*

endswitch;
```

(2) 説明

1. 式が評価されます。
2. 式の値と等しい case ラベル値を持つブロックに制御が移ります。break に達するまでは，（case ラベル値とは関係なく）引き続きすべてのステートメントが実行されます。
3. break ステートメントに達すると，switch ステートメントの外側に制御が移ります。
4. 式の値が case ラベル値と等しくないときは，default ラベルが指定されていると，その位置に制御が移ります。

switch-endswitch ステートメントを指定するには，次の規則に従います。

- switch ステートメントの式のデータ型は，定数値のようなラベル定数のものと同一にします。
- ネストする switch ステートメントの数は，255 以内に抑えてください。
- すべての case ラベルが，関連づけられた実行ステートメントを持つ必要はありませんが，最終の case ステートメントは，関連づけられた実行ステートメントを最低一つ持つようにします。
- switch-endswitch ステートメントは，最大 255 の case ラベルを持てます。
- case ラベルに指定できるのは，数値定数，string 定数，または AIT 言語のマクロで，式は指定できません。

  例

```
case -5:         // 有効
case +6:         // 有効
case "String":   // 有効
case intvar:     // 無効
case 3+2:        // 無効
```

- case ラベルで指定する定数は，switch ステートメントの式と同じ型にします。

  例

```
switch (Stringvar1+Stringvar2)  // 変数はどちらもstring型
case 1:                         // 無効
case "キャプション1":           // 有効
..
..
endswitch;
```

- switch case ステートメントとともに case ステートメントを指定しなければ，文法エラーとして解釈されます。
- switch case ステートメント内に記述できる break ステートメントは，255 個までです。
- switch case ステートメント内の case ステートメントに別のステートメントを指定する必要はありませんが，最終の case ステートメントには，必ず別のステートメントを追加してください。

  例

```
switch(i)
{
  case 1:
  case 2:
    a=b+c; // "a=b+c;"ステートメントを指定しなければ,
           // スクリプトアナライザーから文法エラーが発行される
}
```

(3) 記述例

```
s1="abcdefghijk";
switch (!AIT_IsEmpty(s1 ))
case true:            // s1が空の場合実行されます
    s2 = AIT_StrUpper(s1);
    AIT_MessageBox("s2",s2);
```

```
       if ( ( length = AIT_StrLength(s2)) > 10)
            break;
    endif;
    break;
default:              //状態がfalseを返してきた場合
    break;            //実行されます
endswitch;
```

3.7 関数呼び出し

AIT 言語は，標準的な操作を実行するための各種 API をサポートしています。API は，大きく次のように分類されます。

- ウィンドウ操作
- 確認操作
- 解像度のチェック
- 日時操作
- IME 操作
- 文字列操作
- メッセージ操作
- レジストリ操作
- ディレクトリ操作
- ファイル操作
- INI ファイル操作
- レコーダ操作
- タスクバー操作
- ユティリティ操作
- JP1/NETM/DM とのインターフェース

(1) 形式

● 上記の API は，MAIN および ERROR セクションから呼び出せます。

● パラメタのデータ型は，API 仕様で定義されているものと同一にします。

● 関数呼び出しの中に式を指定できます。また，式のデータ型または関数呼び出しが API 仕様と一致していれば，関数呼び出し同士をネストすることもできます。

● API の実行によってランタイムエラーが発生した場合は，ERROR セクションに制御が移ります。

● 関数の処理が失敗した場合は，関数の戻り値は AIT_GetLastError を呼び出すことで取得できます。エラーメッセージに対応するエラーテキストも AIT_GetErrorText を使用して取得できます。

(2) 記述例

```
integer intvar1;
string Caption;
string Stringvar1, Stringvar2;
...
...
AIT_LogMessage("SAMPLE FUNCTION CALL");  // 関数呼び出し
if(AIT_FocusWindow("Installable-Setup","#32770"))
                                  // 式内で関数呼び出しの戻り値を使用している
   AIT_LogMessage("INSIDE Installable Software SETUP");  // 実行される
   AIT_PlayKey("{Enter}");
endif;
                        // 関数呼び出しの異常終了をチェック
Caption = "Installable Software";
intvar1    = AIT_GetSubStr(Stringvar1, Stringvar2, 50);
                                                    // intvar1が0になる
if(!intvar1)
          AIT_LogMessage(AIT_GetErrorText(AIT_GetLastError()));
endif;
```

3.8 キーワード

キーワードは，特別な意味を持つ定義済みの予約識別子です。キーワードをプログラム内で識別子として使うことはできません。識別子のスペルをキーワードと同じにしてはいけません。キーワードは，大文字小文字を区別しないため，大文字小文字の組み合わせを変えても識別子として使用することはできません。

AIT 言語で定義されているキーワードを表 3-3 に示します。

表 3-3 AIT 言語で定義されているキーワード

項番	キーワード	項番	キーワード	項番	キーワード
1	bool	2	break	3	case
4	const	5	continue	6	default
7	define	8	do	9	ECompany
10	else	11	endif	12	endswitch
13	ERROR	14	EUser	15	float
16	for	17	goto	18	IconGroupName
19	if	20	InstallDirectory	21	InstallDrive
22	InstallerName	23	integer	24	JCompany
25	JUser	26	loop	27	MAIN
28	next	29	PACKAGE_INFO	30	PackageID
31	Product	32	ScriptFileVersion	33	SerialNumber
34	string	35	switch	36	Version
37	while	－	－	－	－

（凡例） －：なし

キーワードには，上記のほかに，API 名，AIT 言語の定義済みマクロ，および幾つかの Win 32 エラーコードがあります。

3.9 マクロ

AIT スクリプト言語には，値を代入できない特定の定義済み定数があります。この種の定数は，次に示すマクロ名とそのデータ型で分類されます。

なお，DEFINE セクションでマクロの宣言はできません。

3.9.1 ウィンドウ操作および確認操作に関するマクロ

ウィンドウ操作および確認操作に関するマクロを表 3-4 に示します。これらのマクロのデータ型は，すべて integer 型です。

表 3-4 ウィンドウ操作および確認操作に関するマクロの一覧

マクロ名	
ALT_OFF	ALT_ON
BUTTON_CTRL	CALENDAR_CTRL
CAPSLOCK	CHECKBOX_CTRL
COMBO_CTRL	COMMANDBUTTON_CTRL
CONTROL_CAPTION_SIZE	CONTROL_CLASS_SIZE
CTRL_OFF	CTRL_ON
DTPICKER_CTRL	EDIT_CTRL
HSCROLL	IDABORT
IDCANCEL	IDIGNORE
IDNO	IDOK
IDRETRY	IDYES
INSERTLOCK	IPADDRESS_CTRL
KEYSTATE_OFF	KEYSTATE_ON
LBUTTON	LISTBOX_CTRL
LIST_CTRL	MBUTTON
MB_ABORTRETRYIGNORE	MB_ICONEXCLAMATION
MB_ICONINFORMATION	MB_ICONQUESTION
MB_ICONSTOP	MB_OK
MB_OKCANCEL	MB_RETRYCANCEL
MB_YESNO	MB_YESNOCANCEL
NOTIFICATION_CTRL	NUMLOCK
OPTIONBUTTON_CTRL	PROGRESS_CTRL
RBUTTON	REBAR_CTRL
SB_BOTTOM	SB_LEFT
SB_LINEDOWN	SB_LINELEFT
SB_LINERIGHT	SB_LINEUP
SB_PAGEDOWN	SB_PAGELEFT
SB_PAGERIGHT	SB_PAGEUP
SB_RIGHT	SB_THUMBPOSITION

マクロ名	
SB_THUMBTRACK	SB_TOP
SCROLLBAR_CTRL	SCROLLLOCK
SHIFT_OFF	SHIFT_ON
SLIDER_CTRL	SPIN_CTRL
STARTBUTTON_CTRL	STATIC_CTRL
STATUSBAR_CTRL	STYLE_CHECK_BUTTON
STYLE_PUSH_BUTTON	STYLE_RADIO_BUTTON
SW_HIDE	SW_MAXIMIZE
SW_MINIMIZE	SW_NORMAL
SW_SHOW	SW_SHOWMAXIMIZED
SW_SHOWMINIMIZED	SW_SHOWMINNOACTIVE
SW_SHOWNOACTIVATE	SW_SHOWNORMAL
TAB_CTRL	TASKBARCLOCK_CTRL
TASKBARITEMS_CTRL	TASKBAR_CTRL
TOOLBAR_CTRL	TOOLTIPS_CTRL
TREE_CTRL	VSCROLL

3.9.2 メッセージ操作に関するマクロ

メッセージ操作に関するマクロを次に示します。このマクロのデータ型は integer 型です。

● HWND_BROADCAST

3.9.3 ファイル操作に関するマクロ

ファイル操作に関するマクロを表 3-5 に示します。これらのマクロのデータ型は，すべて integer 型です。

表 3-5 ファイル操作に関するマクロの一覧

マクロ名	
AIT_ALLFILES	CREATE_ALWAYS
CREATE_NEW	FILE_ALL
FILE_ATTRIBUTE_ARCHIVE	FILE_ATTRIBUTE_DIRECTORY
FILE_ATTRIBUTE_HIDDEN	FILE_ATTRIBUTE_READONLY
FILE_ATTRIBUTE_SYSTEM	FILE_READ
FILE_WRITE	GENERIC_READ
GENERIC_WRITE	OPEN_ALWAYS
OPEN_EXISTING	TRUNCATE_EXISTING

3.9.4 IME 操作に関するマクロ

IME 操作に関するマクロを表 3-6 に示します。これらのマクロのデータ型は，すべて integer 型です。

表 3-6 IME 操作に関するマクロの一覧

マクロ名	
IGP_CONVERSION	IGP_GETIMEVERSION
IGP_PROPERTY	IGP_SELECT
IGP_SENTENCE	IGP_SETCOMPSTR
IGP_UI	IME_CHOTKEY_IME_NONIME_TOGGLE
IME_CHOTKEY_SHAPE_TOGGLE	IME_CHOTKEY_SYMBOL_TOGGLE
IME_CMODE_CHARCODE	IME_CMODE_EUDC
IME_CMODE_FULLSHAPE	IME_CMODE_HANJACONVERT
IME_CMODE_KATAKANA	IME_CMODE_NATIVE
IME_CMODE_NOCONVERSION	IME_CMODE_ROMAN
IME_CMODE_SOFTKBD	IME_CMODE_SYMBOL
IME_JHOTKEY_CLOSE_OPEN	IME_KHOTKEY_ENGLISH
IME_KHOTKEY_HANJACONVERT	IME_KHOTKEY_SHAPE_TOGGLE
IME_PROP_AT_CARET	IME_PROP_CANDLIST_START_FROM_1
IME_PROP_SPECIAL_UI	IME_PROP_UNICODE
IME_SMODE_AUTOMATIC	IME_SMODE_CONVERSATION
IME_SMODE_NONE	IME_SMODE_PHRASEPREDICT
IME_SMODE_PLAURALCLAUSE	IME_SMODE_PLURALCLAUSE
IME_SMODE_SINGLECONVERT	IME_THOTKEY_IME_NONIME_TOGGLE
IME_THOTKEY_SHAPE_TOGGLE	IME_THOTKEY_SYMBOL_TOGGLE
SCS_CAP_COMPSTR	SCS_CAP_MAKEREAD
SELECT_CAP_CONVERSION	SELECT_CAP_SENTENCE
UI_CAP_2700	UI_CAP_ROT90
UI_CAP_ROTANY	―

（凡例） ―：なし

3.9.5 ユティリティ操作に関するマクロ

ユティリティ操作に関するマクロを表 3-7 に示します。これらのマクロのデータ型は，すべて integer 型です。

表 3-7 ユティリティ操作に関するマクロの一覧

マクロ名	
VER_PLATFORM_WIN32_NT	VER_PLATFORM_WIN32_WINDOWS

3.9.6 レジストリ操作に関するマクロ

レジストリ操作に関するマクロを表 3-8 に示します。これらのマクロのデータ型は，すべて integer 型です。

表 3-8 レジストリ操作に関するマクロの一覧

マクロ名	
HKEY_CLASSES_ROOT	HKEY_CURRENT_CONFIG
HKEY_CURRENT_USER	HKEY_LOCAL_MACHINE
HKEY_USERS	―

（凡例） ―：なし

3.9.7 ディレクトリ操作に関するマクロ

ディレクトリ操作に関するマクロを表 3-9 に示します。これらのマクロのデータ型は，すべて integer 型です。

表 3-9 ディレクトリ操作に関するマクロの一覧

マクロ名	
AIT_CURRENTDIRECTORY	AIT_PARENTDIRECTORY

3.9.8 エラーロギングに関するマクロ

エラーロギングに関するマクロを表 3-10 に示します。これらのマクロのデータ型は，すべて integer 型です。

表 3-10 エラーロギングに関するマクロの一覧

マクロ名	
ERROR_ACCESS_DENIED	ERROR_ALREADY_EXISTS
ERROR_BADDB	ERROR_BADKEY
ERROR_BAD_COMMAND	ERROR_BAD_NETPATH
ERROR_BAD_NET_NAME	ERROR_BAD_PATHNAME
ERROR_BUFFER_OVERFLOW	ERROR_CANNOT_MAKE
ERROR_CANTOPEN	ERROR_CANTREAD
ERROR_CANTWRITE	ERROR_CHILD_MUST_BE_VOLATILE
ERROR_CONTROL_ID_NOT_FOUND	ERROR_CRC
ERROR_CURRENT_DIRECTORY	ERROR_DIRECTORY
ERROR_DISK_CORRUPT	ERROR_DISK_FULL
ERROR_FILENAME_EXCED_RANGE	ERROR_FILE_CORRUPT
ERROR_FILE_EXISTS	ERROR_FILE_NOT_FOUND
ERROR_HANDLE_DISK_FULL	ERROR_HANDLE_EOF
ERROR_INSUFFICIENT_BUFFER	ERROR_INVALID_COMPUTERNAME
ERROR_INVALID_DATA	ERROR_INVALID_DRIVE
ERROR_INVALID_FUNCTION	ERROR_INVALID_HANDLE
ERROR_INVALID_INDEX	ERROR_INVALID_MENU_HANDLE
ERROR_INVALID_NAME	ERROR_INVALID_NETNAME
ERROR_INVALID_PARAMETER	ERROR_INVALID_SCROLLBAR_RANGE
ERROR_INVALID_SHARENAME	ERROR_INVALID_WINDOW_HANDLE

マクロ名	
ERROR_KEY_DELETED	ERROR_KEY_HAS_CHILDREN
ERROR_LOCK_VIOLATION	ERROR_MENU_ITEM_NOT_FOUND
ERROR_NETWORK_BUSY	ERROR_NETWORK_UNREACHABLE
ERROR_NOACCESS	ERROR_NOT_ENOUGH_MEMORY
ERROR_NOT_READY	ERROR_NOT_REGISTRY_FILE
ERROR_NO_LOG_SPACE	ERROR_NO_MORE_FILES
ERROR_NO_MORE_ITEMS	ERROR_NO_MORE_SEARCH_HANDLES
ERROR_NO_SCROLLBARS	ERROR_OUTOFMEMORY
ERROR_PATH_BUSY	ERROR_PATH_NOT_FOUND
ERROR_READ_FAULT	ERROR_REGISTRY_CORRUPT
ERROR_REGISTRY_IO_FAILED	ERROR_REGISTRY_RECOVERED
ERROR_SHARING_VIOLATION	ERROR_SUCCESS
ERROR_TIMEOUT	ERROR_TOO_MANY_OPEN_FILES
ERROR_WRITE_FAULT	ERROR_WRITE_PROTECT

4 API リファレンス

この章では，AIT 言語で使用する API について説明します。

4.1 API 一覧

AIT 言語が提供する API は，以下のカテゴリで分類されます。

- ウィンドウ操作
- 確認操作
- 解像度のチェック
- 日時操作
- IME 操作
- 文字列操作
- メッセージ操作
- レジストリ操作
- ディレクトリ操作
- ファイル操作
- INI ファイル操作
- レコーダ操作
- タスクバー操作
- ユティリティ操作
- JP1/NETM/DM とのインターフェース

カテゴリ別の API の一覧を次の項以降に示します。

4.1.1 ウィンドウ操作

次の API は，ウィンドウの検索，特定のコントロールへのフォーカス設定，チェックボックスまたはラジオボタンのチェック状態の設定など，ウィンドウおよびコントロールに関連する処理をアプリケーションが実行できるようにします。

API 関数名	機能
AIT_FocusWindow	ウィンドウを検索し，フォーカスを設定します。
AIT_ExistWindow	ウィンドウが存在するかをチェックします。
AIT_MinWnd	ウィンドウを最小化します。
AIT_SetWndPos	指定したウィンドウの位置を変更します。
AIT_SetWndPosSize	指定したウィンドウの位置とサイズを変更します。
AIT_GetWindowText	指定したウィンドウのタイトルを取得します。
AIT_SetActWnd	アクティブなウィンドウを設定します。
AIT_GetCtrlText	ウィンドウのコントロールからテキストを取得します。
AIT_CtrlSetFocus	コントロールにフォーカスを設定します。
AIT_SetSpinPos	スピンまたはスライダーコントロールの位置を設定します。
AIT_SetScrollPos	スクロールバーの位置を設定します。
AIT_CtrlClick	コントロールに対しマウスクリック処理を実行します。
AIT_SelectMultipleListItem	複数リスト項目を選択します。
AIT_SelectListItem	リスト項目を選択します。
AIT_SelectIPAddressField	IP アドレスフィールドを選択します。
AIT_SelectText	コントロール上のテキストを選択します。
AIT_DefaultButtonCount	デフォルトボタンスタイルの数を取得します。

API 関数名	機能
AIT_SetCheck	ラジオボタンまたはチェックボックスのチェック状態を設定します。
AIT_CtrlItemCount	コントロールの項目数を取得します。
AIT_GetIndexText	インデックスからテキストを取得します。
AIT_CtrlItemIndex	テキストのインデックスを取得します。
AIT_GetIndexTextLen	基準値 0 のインデックス文字列の，文字の長さを取得します。
AIT_SetKeyState	キーの状態を設定します。
AIT_GetKeyState	キー状態を取得します。
AIT_MouseClick	マウスクリック操作を実行します。
AIT_MouseUp	マウスを離す操作を実行します。
AIT_MouseDown	マウスを押す操作を実行します。
AIT_MouseMoveTo	マウスの移動操作を実行します。
AIT_MouseDragDrop	マウスのドラッグアンドドロップ操作を実行します。
AIT_MouseDblClk	マウスのダブルクリック操作を実行します。
AIT_SetComboEditSelText	コンボボックスのテキストを選択します。
AIT_ComboBoxCloseUp	コンボボックスへのクローズアップ操作を実行します。
AIT_ComboBoxDropDown	コンボボックスへのドロップダウン操作を実行します。
AIT_GetEditFirstLineIndex	エディットボックスでの最初の行のインデックスを取得します。
AIT_GetEditCurrentLineIndex	エディットボックスでの現在の行のインデックスを取得します。
AIT_GetCtrlTextLen	コントロールのテキストの長さを取得します。
AIT_GetEditTextLineLen	エディットボックスにでの行の長さを取得します。
AIT_GetDtPickerTime	日時ピッカーから時間を取得します。
AIT_GetDtPickerDate	日時ピッカーから日付を取得します。
AIT_SetDtPickerTime	日時ピッカーで時間を設定します。
AIT_SetDtPickerDate	日時ピッカーで日付を設定します。
AIT_GetMenu	メニューハンドルを取得します。
AIT_GetSubMenu	サブメニューハンドルを取得します。
AIT_GetMenuText	メニューテキストを取得します。
AIT_GetMenuIndex	メニューのインデックスを取得します。
AIT_MenuItemClick	指定したメニュー項目をクリックします。

4.1.2 確認操作

次の API は，コントロールの存在，コントロールの状態，コントロールのフォーカスを確認するなど，確認処理をアプリケーションが実行できるようにします。

API 関数名	機能
AIT_VerifyExistence	コントロールの存在を確認します。
AIT_VerifyEnabled	コントロールが使用できるかどうかを確認します。
AIT_VerifyFocus	コントロールがフォーカスされているかどうかを確認します。
AIT_VerifyState	コントロールの状態を確認します。
AIT_VerifyCharPos	コントロールの文字の位置を確認します。

API 関数名	機能
AIT_VerifyLine	行のインデックスを確認します。
AIT_VerifyText	指定したテキストを確認します。
AIT_VerifySelected	エディットボックスで選択されたテキストを確認します。
AIT_VerifyFirstVisible	コントロールの最初の可視項目を確認します。
AIT_VerifyCount	項目カウントを確認します。
AIT_VerifyPos	コントロールの位置を確認します。
AIT_VerifyIndex	コントロールのインデックスを確認します。
AIT_VerifyLocation	コントロールの位置を確認します。
AIT_VerifyDateTime	コントロールの日付または時間の値を確認します。
AIT_VerifyKeyState	キーの状態を確認します。
AIT_VerifyDefaultButton	デフォルトのボタンスタイルを確認します。
AIT_VerifyNoOfCtrls	ウィンドウのコントロール数を確認します。
AIT_VerifyMenuChecked	メニューのチェック状態を確認します。
AIT_VerifyMenuEnabled	メニューの使用可能状態を確認します。

4.1.3 解像度のチェック

次の API は，システムの解像度をチェックします。

API 関数名	機能
AIT_CheckResolution	スクリーンの解像度をチェックします。

4.1.4 日時操作

次の API は，アプリケーションが日付または時間を処理できるようにします。

API 関数名	機能
AIT_GetDate	システムの日付を取得します。
AIT_GetTime	システムの時間を取得します。

4.1.5 IME 操作

次の API は，IME のオンまたはオフ，変換モードの切り替えなど，ユーザがよく実行する操作をアプリケーションがシミュレートできるようにします。

API 関数名	機能
AIT_IMEGetOpenStatus	IME がオープンステータスを取得します。
AIT_IMESetOpenStatus	IME がオープンステータスを設定します。
AIT_IMEGetConversionStatus	IME が変換ステータスを取得します。
AIT_IMESetConversionStatus	IME が変換ステータスを設定します
AIT_IMEGetStatusWindowPos	IME がステータスウィンドウの位置を取得します。
AIT_IMESetStatusWindowPos	IME がステータスウィンドウの位置を設定します。

API 関数名	機能
AIT_IMEGetProperty	IME がプロパティを取得します。
AIT_IMESimulateHotKey	IME がホットキーをシミュレートします。

4.1.6 文字列操作

次の API は，部分文字列処理，切り取り処理，ASCII コードへの変換など，文字列処理をアプリケーションが実行できるようにします。

API 関数名	機能
AIT_GetSubStr	文字列から指定した長さの文字列を返します。
AIT_FindSubStr	文字列を検索し，一致する最初の文字列の位置を返します。
AIT_StrLength	文字列の長さを取得します。
AIT_IsEmpty	文字列が空かどうかを確認します。
AIT_StrLTrim	文字列から左端文字を切り取ります。
AIT_StrRTrim	文字列から右端文字を切り取ります。
AIT_StrTrim	文字列から文字を切り取ります。
AIT_StrUpper	文字列を大文字に変換します。
AIT_StrLower	文字列を小文字に変換します。
AIT_StrLeft	文字列の左から指定した文字数を取得します。
AIT_StrRight	文字列の右から指定した文字数を取得します。
AIT_CharToASCII	文字を ASCII コードに変換します。
AIT_ASCIIToChar	ASCII コードから文字に変換します。

4.1.7 メッセージ操作

次の API は，新規のウィンドウメッセージを記録し，ほかのウィンドウにメッセージを記入します。

API 関数名	機能
AIT_RegisterWindowMessage	ウィンドウメッセージを記録します。
AIT_PostMessage	メッセージを記入します。

4.1.8 レジストリ操作

次の API は，レジストリキーの作成，キー値の照会，キーの削除など，アプリケーションがレジストリを処理できるようにします。

API 関数名	機能
AIT_RegCreateKey	レジストリキーを作成します。
AIT_RegDeleteKey	レジストリキーを削除します。
AIT_RegDeleteValue	レジストリ値を削除します。
AIT_RegOpenKey	レジストリキーを開きます。
AIT_RegCloseKey	レジストリキーを閉じます。

API 関数名	機能
AIT_RegGetStringValue	レジストリの string 値を取得します。
AIT_RegGetDWORDValue	レジストリの DWORD 値を取得します。
AIT_RegSetStringValue	レジストリの string 値を設定します。
AIT_RegSetDWORDValue	レジストリの DWORD 値を設定します。
AIT_RegKeyExists	レジストリキーが存在するかどうかを確認します。
AIT_RegValueExists	レジストリ値が存在するかどうかを確認します。

4.1.9 ディレクトリ操作

次の API は，アプリケーションがディレクトリを作成，削除，およびコピーすることを可能にします。

API 関数名	機能
AIT_DirCreate	ディレクトリを作成します。
AIT_DirRemove	現在のディレクトリを削除します。
AIT_DirCopy	ディレクトリをコピーします。
AIT_SetCurrentDirectory	現在のディレクトリを設定します。
AIT_GetCurrentDirectory	現在のディレクトリを取得します。

4.1.10 ファイル操作

次の API は，アプリケーションがファイルを作成，コピー，削除およびリネームすることを可能にします。

API 関数名	機能
AIT_FileOpen	ファイルを開くか作成します。
AIT_FileClose	ファイルを閉じます。
AIT_FileGetLine	ファイルからデータを取得します。
AIT_FilePutLine	ファイルにデータを書き込みます。
AIT_FileGetPos	ファイルポインタの位置を取得します。
AIT_FileSetPos	ファイルポインタの位置を設定します。
AIT_FileEOF	ファイルの終わりであるかをチェックします。
AIT_FileSize	ファイルのサイズを取得します。
AIT_FileCopy	ファイルをコピーします。
AIT_FileDelete	ファイルを削除します。
AIT_FileExists	ファイルの存在をチェックします。
AIT_FileRename	ファイルをリネームします。
AIT_ChangeFileAttribute	ファイルの属性を変更します。
AIT_FindFirstFile	初めのファイルを検索します。
AIT_FindNextFile	次のファイルを検索します。
AIT_FindCloseFile	検索ハンドルを閉じます。

4.1.11 INI ファイル操作

次の API は，セクションでのキー値を取得する，セクションのすべての内容を取得するなどの初期化（INI）ファイル処理をアプリケーションが実行できるようにします。

API 関数名	機能
AIT_GetProfileString	プロファイル string を取得します。
AIT_SetProfileString	プロファイル string を設定します。
AIT_GetProfileFirstSection	最初のセクションを取得します。
AIT_GetProfileNextSection	次のプロファイルセクションを取得します。
AIT_GetProfileFirstSectionNames	最初のセクション名を取得します。
AIT_GetProfileNextSectionNames	次のセクション名を取得します。

4.1.12 レコーダ操作

次の API は，自動インストールに使う AIT ファイルを作成するための各種 API です。

API 関数名	機能
AIT_Sleep	指定した期間，AIT ファイルの実行を中断します。
AIT_SetDefaultWaitTimeout	デフォルトの待機タイムアウト時間を設定します。
AIT_Exec	指定した処理を実行します。
AIT_ExecCommand	指定した MS-DOS コマンドまたはシステムコマンドを実行します。
AIT_PlayKey	アクティブなウィンドウへキーボード入力情報を送信します。

4.1.13 タスクバー操作

次の API は，タスクバー中の項目に対するクリックや，タスクバーがフォーカスを持っているかの確認などの，タスクバーに関する操作をするための API です。

API 関数名	機能
AIT_TaskbarItemExists	タスクバーに特定の項目が存在するかどうかをチェックします。
AIT_TaskbarItemIndex	タスクバー上の選択されている項目の（1 を基準とした）インデックスを検索します。
AIT_TaskbarItemClk	シングルクリックの働きをするマウスボタンを，タスクバーの特定のタブ項目に設定します。
AIT_TaskbarItemSelected	タスクバーの特定の項目が選択されているかどうかをチェックします。
AIT_TaskbarClk	シングルクリックと同等の働きをするマウスボタンを，タスクバーの空いている場所に設定します。
AIT_TaskbarHasFocus	タスクバーが入力フォーカスを持っているかどうかをチェックします。
AIT_TaskbarSetFocus	タスクバーに入力フォーカスを設定します。

4.1.14 ユティリティ操作

次の API は，メッセージボックスやステータスボックスの表示，OS タイプの取得などをアプリケーションが処理できるようにするユティリティ API です。

API 関数名	機能
AIT_MessageBox	メッセージボックスを表示します。
AIT_StatusBox	ステータスボックスを表示します。
AIT_StatusBoxClose	ステータスボックスを閉じます。
AIT_GetEnv	環境変数の内容を取得します。
AIT_GetLastError	最後に発生したエラーを取得します。
AIT_GetErrorText	エラーメッセージを取得します。
AIT_InitLog	ログファイルを初期化します。
AIT_LogMessage	ログファイルにメッセージを記録します。
AIT_Exit	AIT ファイルの処理を終了します。
AIT_GetOSType	OS タイプを取得します。

4.1.15 JP1/NETM/DM とのインターフェース

次の API は，JP1/NETM/DM とのインターフェースを使用できるようにする API です。

API 関数名	機能
AIT_DMPSTRC	JP1/NETM/DM に固有のグローバル変数を設定します。 JP1/NETM/DM で AIT ファイルを使用したリモートインストールを実行する場合は，必ず指定します。

4.2 API の詳細

ここでは，各 API の詳細を説明します。

API の説明形式

API は，基本的には次に示す形式で説明しています。各 API は，API 名のアルファベット順に並んでいます。

機能

API の機能を示しています。

形式

API の記述形式を示しています。

[ ] で囲まれた引数は省略できます。省略した場合は，デフォルト値が設定されます。これらの引数は常に API の最後の引数となります。

（例 1）

```
bool AIT_SelectListItem ( strCaption, nCtrlType, strItemText [, fTimeOut] );
```

上記の API では，fTimeOut は関数呼び出し時に指定する必要はありません。

（例 2）

```
AIT_SelectListItem ( "Countries" LISTBOX_CTRL, "Japan" );
integer AIT_MessageBox ( strMessage, strTitle [, nIconType] [, nMsgBoxType] );
```

上記の API では，nIconType と nMsgBoxType の両方を省略して次のように呼び出せます。

```
AIT_MessageBox( "Hello World","Error");
```

nMsgBoxType を指定して nIconType を指定したくない場合，nIconType のデフォルト値が設定されます。

```
AIT_MessageBox( "Hello World","Error", MB_ICONEXCLAMATION );
```

引数

API で指定できる引数を示しています。入力用の引数の場合は（入力用），出力用の引数の場合は（出力用），入力用で省略できる場合は（入力用，省略可）と示しています。

戻り値

API の戻り値を示しています。

API が実行中に処理を失敗した場合のためにエラーコードが設定されています。このエラーコードは AIT_GetLastError という API を使用して取得できます。ユーザが特殊なコードのために特殊なエラー処理を必要とする場合は，API が正常に実行されたか失敗したかをチェックしたあとで，AIT ファイルによって適切に扱うことになります。

ランタイムエラーが発生した場合，AIT ファイル内の ERROR セクションが自動的に実行されます。

注意事項

API 実行時の注意事項がある場合に記載しています。

AIT_ASCIIToChar

機能

指定した ASCII コードを文字に変換します。

形式

```
string AIT_ASCIIToChar  (
  integer nASCIIValue     // ASCIIコード
);
```

引数

nASCIIValue**（入力用）**

ASCII コードを指定してください。

戻り値

ASCII コードに対応する文字が返されます。

AIT_ChangeFileAttribute

機能

指定したファイルまたはディレクトリの属性を変更します。

形式

```
bool AIT_ChangeFileAttribute  (
  string strFileName,     // ファイル名
  integer nFileAttributes // ファイル属性
);
```

引数

strFileName**（入力用）**

ファイル名を指定してください。ディレクトリを指定することもできます。

nFileAttributes**（入力用）**

変更後の属性を指定してください。入力に使用できる値については,「AIT_FileExists」を参照してください。

戻り値

この関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得してください。

関数が正常に実行されなかった場合に AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
2	ERROR_FILE_NOT_FOUND
3	ERROR_PATH_NOT_FOUND
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
18	ERROR_NO_MORE_FILES
21	ERROR_NOT_READY
53	ERROR_BAD_NETPATH
87	ERROR_INVALID_PARAMETER
123	ERROR_INVALID_NAME
148	ERROR_PATH_BUSY
161	ERROR_BAD_PATHNAME
183	ERROR_ALREADY_EXISTS
206	ERROR_FILENAME_EXCED_RANGE
1005	ERROR_UNRECOGNIZED_VOLUME
1210	ERROR_INVALID_COMPUTERNAME
1214	ERROR_INVALID_NETNAME
1231	ERROR_NETWORK_UNREACHABLE
1392	ERROR_FILE_CORRUPT

AIT_CharToASCII

機能

指定した文字を ASCII コードに変換します。

形式

```
integer AIT_CharToASCII (
  string strStrName     // 1文字以上の文字列
);
```

引数

strStrName**（入力用）**

文字列名を指定してください。

戻り値

最初の文字の ASCII コードが返されます。

AIT_CheckResolution

機能

指定した解像度が現在の画面の解像度と適合しているかどうかをチェックします。

形式

```
integer AIT_CheckResolution  (
  integer nWidth,     // チェックする画面の幅
  integer nHeight     // チェックする画面の高さ
);
```

引数

nWidth**（入力用）**

プライマリディスプレイモニタの画面でチェックする幅をピクセル単位で指定してください。

nHeight**（入力用）**

プライマリディスプレイモニタの画面でチェックする高さをピクセル単位で指定してください。

戻り値

指定した解像度が現在の解像度と一致する場合，戻り値は 1 です。指定した解像度が現在の解像度と一致しない場合，戻り値は 0 です。失敗した場合，戻り値は -1 です。-1 が返された場合は，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
87	ERROR_INVALID_PARAMETER

AIT_ComboBoxCloseUp

機能

コンボボックスのクローズアップをシミュレートします。

形式

```
bool AIT_ComboBoxCloseUp  (
  string strCaption    // コントロールのキャプション
  [,float fTimeOut]    // タイムアウト時間
);
bool AIT_ComboBoxCloseUp  (
  integer nCtrlID     // コントロールID
  [,float fTimeOut]   // タイムアウト時間
);
```

引数

strCaption**（入力用）**

コントロールのキャプションを指定してください。

nCtrlID**（入力用）**

コントロール ID を指定してください。

fTimeOut**（入力用，省略可）**

コントロールが戻らなかった場合に再試行するためのタイムアウト値を秒単位で指定してください。この関数がコントロールを見つけるために使用できる最大時間を秒単位で指定してください。省略した場合は，関数 AIT_SetDefaultWaitTimeout で設定した値が使用されます。

戻り値

この関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

注意事項

コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連するラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_ComboBoxDropDown

機能

コンボボックスのドロップダウンをシミュレートします。

形式

```
bool AIT_ComboBoxDropDown  (
  string strCaption       // コントロールのキャプション
  [,float fTimeOut]       // タイムアウト時間
);
bool AIT_ComboBoxDropDown  (
  integer nCtrlID         // コントロールID
  [,float fTimeOut]       // タイムアウト時間
);
```

引数

strCaption（入力用）

コントロールのキャプションを指定してください。

nCtrlID（入力用）

コントロール ID を指定してください。

fTimeOut（入力用，省略可）

この関数がコントロールを見つけるために使用できる最大時間を秒単位で指定してください。省略した場合は，関数 AIT_SetDefaultWaitTimeout で設定した値が使用されます。

戻り値

この関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

注意事項

コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連するラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_CtrlClick

機能

アクティブなウィンドウの特定のコントロールでマウスをクリックします。

形式

```
bool AIT_CtrlClick  (
  string strCaption,      // コントロールのキャプション
  integer nCtrlType,      // コントロールタイプ
  integer nMouseButton    // マウスボタン
  [,float fTimeOut]       // タイムアウト時間
);
bool AIT_CtrlClick  (
  integer nCtrlID,        // コントロールID
  integer nCtrlType,      // コントロールタイプ
  integer nMouseButton    // マウスボタン
  [,float fTimeOut]       // タイムアウト時間
);
```

引数

strCaption**（入力用）**

コントロールのキャプションを指定してください。

nCtrlID**（入力用）**

コントロール ID を指定してください。

nCtrlType**（入力用）**

コントロールタイプを指定してください。これは，次の値のどれかにする必要があります。

値	意味
BUTTON_CTRL	コントロールタイプはコマンドボタンです。
CHECKBOX_CTRL	コントロールタイプはチェックボックスです。

値	意味
OPTIONBUTTON_CTRL	コントロールタイプはオプションボタンです。
EDIT_CTRL	コントロールタイプはエディットボックスです。
STATIC_CTRL	コントロールタイプはスタティックテキストコントロールです。
COMBO_CTRL	コントロールタイプはコンボボックスです。
LISTBOX_CTRL	コントロールタイプはリストボックスです。
SPIN_CTRL	コントロールタイプはスピンコントロールです。
TREE_CTRL	コントロールタイプはツリーコントロールです。
LIST_CTRL	コントロールタイプはリストコントロールです。
DTPICKER_CTRL	コントロールタイプは日時ピッカーです。

nMouseButton**（入力用，省略可）**

マウスでクリックするボタンを指定してください。次の値のどれかにする必要があります。

値	意味
LBUTTON	左マウスボタン
MBUTTON	中央マウスボタン
RBUTTON	右マウスボタン

省略した場合は，LBUTTON がデフォルト値として使用されます。

fTimeOut**（入力用，省略可）**

この関数がコントロールを見つけるために使用できる最大時間を秒単位で指定してください。省略した場合は，関数 AIT_SetDefaultWaitTimeout で設定した値が使用されます。

戻り値

この関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

注意事項

コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連するラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_CtrlItemCount

機能

アクティブなウィンドウの特定のコントロールで，項目数を取得します。

形式

```
bool AIT_CtrlItemCount  (
  string strCaption,     // コントロールのキャプション
  integer nCtrlType,     // コントロールタイプ
  integer nItemCount     // 項目数
  [,float fTimeOut]      // タイムアウト時間
);
bool AIT_CtrlItemCount  (
  integer nCtrlID,       // コントロールID
  integer nCtrlType,     // コントロールタイプ
  integer nItemCount     // 項目のカウント
  [,float fTimeOut]      // タイムアウト時間
);
```

引数

strCaption（入力用）

コントロールのキャプションを指定してください。

nCtrlID（入力用）

コントロール ID を指定してください。

nCtrlType（入力用）

コントロールタイプを指定してください。これは，次の値のどれかにする必要があります。

値	意味
COMBO_CTRL	コントロールタイプはコンボボックスです。
LISTBOX_CTRL	コントロールタイプはリストボックスです。
TREE_CTRL	コントロールタイプはツリーコントロールです。
LIST_CTRL	コントロールタイプはリストコントロールです。

nItemCount（出力用）

コントロール内の項目数を受け取る変数を指定してください。関数から制御が戻ると，この変数に項目数が格納されます。

fTimeOut（入力用，省略可）

この関数がコントロールを見つけるために使用できる最大時間を秒単位で指定してください。省略した場合は，関数 AIT_SetDefaultWaitTimeout で設定した値が使用されます。

戻り値

この関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
6	ERROR_INVALID_HANDLE
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

注意事項

コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連するラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_CtrlItemIndex

機能

アクティブなウィンドウの特定のコントロールで，項目テキストのインデックスを取得します。

形式

```
bool AIT_CtrlItemIndex  (
  string strCaption,     // コントロールのキャプション
  integer nCtrlType,     // コントロールタイプ
  string strItemText,    // 項目テキスト
  integer nIndex         // 項目インデックス
  [,float fTimeOut]      // タイムアウト時間
);
bool AIT_CtrlItemIndex  (
  integer nCtrlID,       // コントロールID
  integer nCtrlType,     // コントロールタイプ
  string strItemText,    // 項目のテキスト
  integer nIndex         // 項目のインデックス
  [,float fTimeOut]      // タイムアウト時間
);
```

引数

strCaption**（入力用）**

コントロールのキャプションを指定してください。

nCtrlID**（入力用）**

コントロール ID を指定してください。

nCtrlType**（入力用）**

コントロールタイプを指定してください。これは，次の値のどれかにする必要があります。

値	意味
COMBO_CTRL	コントロールタイプはコンボボックスです。
LISTBOX_CTRL	コントロールタイプはリストボックスです。

値	意味
TREE_CTRL	コントロールタイプはツリーコントロールです。
LIST_CTRL	コントロールタイプはリストコントロールです。

strItemText（入力用）

インデックスを取得するための項目テキストを指定してください。

nIndex（出力用）

テキストのインデックスを受け取る変数を指定してください。関数から制御が戻ると，この変数に項目テキストのインデックスが格納されます。インデックスの基準値は 0 です。

fTimeOut（入力用，省略可）

この関数がコントロールを見つけるために使用できる最大時間を秒単位で指定してください。省略した場合は，関数 AIT_SetDefaultWaitTimeout で設定した値が使用されます。

戻り値

この関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
6	ERROR_INVALID_HANDLE
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1413	ERROR_INVALID_INDEX
1460	ERROR_TIMEOUT

注意事項

コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連するラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_CtrlSetFocus

機能

特定のコントロールにフォーカスを設定します。

形式

```
bool AIT_CtrlSetFocus (
  string strCaption,      // コントロールのキャプション
  integer nCtrlType       // コントロールタイプ
```

```
  [,float fTimeOut]      // タイムアウト時間
);
bool AIT_CtrlSetFocus (
  integer nCtrlID,       // コントロールID
  integer nCtrlType      // コントロールタイプ
  [,float fTimeOut]      // タイムアウト時間
);
bool AIT_CtrlSetFocus (
  integer nIndex         // コントロールのインデックス
  [,float fTimeOut]        // タイムアウト時間
);
```

引数

strCaption（入力用）

コントロールのキャプションを指定してください。

nCtrlID（入力用）

コントロール ID を指定してください。

nCtrlType（入力用）

コントロールタイプを指定してください。これは，次の値のどれかにする必要があります。

値	意味
BUTTON_CTRL	コントロールタイプはコマンドボタンです。
CHECKBOX_CTRL	コントロールタイプはチェックボックスです。
OPTIONBUTTON_CTRL	コントロールタイプはオプションボタンです。
EDIT_CTRL	コントロールタイプはエディットボックスです。
STATIC_CTRL	コントロールタイプはスタティックテキストです。
COMBO_CTRL	コントロールタイプはコンボボックスです。
LISTBOX_CTRL	コントロールタイプはリストボックスです。
SPIN_CTRL	コントロールタイプはスピンコントロールです。
TREE_CTRL	コントロールタイプはツリーコントロールです。
LIST_CTRL	コントロールタイプはリストコントロールです。
DTPICKER_CTRL	コントロールタイプは日時ピッカーです。

nIndex（入力可）

コントロールのタブオーダーを指定してください。

fTimeOut（入力用，省略可）

この関数がコントロールを見つけるために使用できる最大時間を秒単位で指定してください。省略した場合は，関数 AIT_SetDefaultWaitTimeout で設定した値が使用されます。

戻り値

この関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

注意事項

コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連するラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_DefaultButtonCount

機能

アクティブなウィンドウの，デフォルトボタンを持つコマンドボタンの数を取得します。

形式

```
integer AIT_DefaultButtonCount ();
```

引数

なし

戻り値

関数が正常に処理された場合，戻り値はデフォルトボタンを持つコマンドボタンの数です。そのほかの場合，戻り値は false です。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。
AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
112	ERROR_DISK_FULL
1460	ERROR_TIMEOUT

AIT_DirCopy

機能

ディレクトリをほかの場所にコピーします。コピー先ディレクトリがすでに存在する場合は，そのディレクトリが上書きされます。

形式

```
bool AIT_DirCopy  (
  string strSourceDirName,      // コピー元ディレクトリ名
  string strTargetDirName       // コピー先ディレクトリ名
);
```

引数

strSourceDirName（入力用）

コピー元ディレクトリ名を指定してください。

strTargetDirName（入力用）

コピー先ディレクトリ名を指定してください。

戻り値

この関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。
AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
87	ERROR_INVALID_PARAMETER
266	ERROR_CANNOT_COPY

AIT_DirCreate

機能

新規ディレクトリを作成します。

形式

```
bool AIT_DirCreate  (
  string strDirName    // ディレクトリ名
);
```

引数

strDirName（入力用）

作成するディレクトリ名を指定してください。

戻り値

この関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。
AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
3	ERROR_PATH_NOT_FOUND
5	ERROR_ACCESS_DENIED
8	ERROR_NOT_ENOUGH_MEMORY

拡張エラー番号	エラーコード
14	ERROR_OUTOFMEMORY
15	ERROR_INVALID_DRIVE
19	ERROR_WRITE_PROTECT
21	ERROR_NOT_READY
23	ERROR_CRC
53	ERROR_BAD_NETPATH
64	ERROR_NETNAME_DELETED
82	ERROR_CANNOT_MAKE
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
123	ERROR_INVALID_NAME
148	ERROR_PATH_BUSY
183	ERROR_ALREADY_EXISTS
206	ERROR_FILENAME_EXCED_RANGE
267	ERROR_DIRECTORY
1005	ERROR_UNRECOGNIZED_VOLUME
1210	ERROR_INVALID_COMPUTERNAME
1214	ERROR_INVALID_NETNAME
1231	ERROR_NETWORK_UNREACHABLE

AIT_DirRemove

機能

既存のディレクトリを削除します。

形式

```
bool AIT_DirRemove  (
  string strDirName    // ディレクトリ名
);
```

引数

strDirName**（入力用）**

ディレクトリ名を指定してください。

戻り値

この関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。
AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
16	ERROR_CURRENT_DIRECTORY
87	ERROR_INVALID_PARAMETER

AIT_DMPSTRC

機能

JP1/NETM/DM に固有のグローバル変数を設定します。JP1/NETM/DM で AIT ファイルを使用したリモートインストールを実行する場合は，必ず指定してください。

この API は，AIT ファイルでインストールプログラムを起動する前に呼び出されます。

形式

```
bool AIT_DMPSTRC ();
```

引数

なし

戻り値

この関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
2	ERROR_FILE_NOT_FOUND
3	ERROR_PATH_NOT_FOUND
5	ERROR_ACCESS_DENIED
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
15	ERROR_INVALID_DRIVE
21	ERROR_NOT_READY
38	ERROR_HANDLE_EOF
53	ERROR_BAD_NETPATH
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
123	ERROR_INVALID_NAME
148	ERROR_PATH_BUSY
161	ERROR_BAD_PATHNAME
206	ERROR_FILENAME_EXCED_RANGE
1005	ERROR_UNRECOGNIZED_VOLUME
1210	ERROR_INVALID_COMPUTERNAME
1214	ERROR_INVALID_NETNAME
1231	ERROR_NETWORK_UNREACHABLE

AIT_Exec

機能

指定したアプリケーションファイルを実行します。

形式

```
bool AIT_Exec  (
  string strExeName,      // アプリケーションファイルの名前
  integer nShowState      // 表示状態
);
```

引数

strExeName（入力用）

アプリケーションファイルの名前を指定してください。

nShowState（入力用）

アプリケーションの表示方法を指定してください。次のどれかにする必要があります。

値	意味
SW_HIDE	アプリケーションを非表示にします。
SW_SHOWNORMAL or SW_NORMAL	アプリケーションを通常の状態で表示します。
SW_SHOWMINIMIZED	アプリケーションを最小化します。
SW_SHOWMAXIMIZED or SW_MAXIMIZE	アプリケーションを最大化します。
SW_SHOWNOACTIVATE	アプリケーションをフォーカスなしで通常の状態で表示します。
SW_SHOWMINNOACTIVE	アプリケーションをフォーカスなしで最小化します。

戻り値

この関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。
AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
2	ERROR_FILE_NOT_FOUND
3	ERROR_PATH_NOT_FOUND
5	ERROR_ACCESS_DENIED
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
21	ERROR_NOT_READY
87	ERROR_INVALID_PARAMETER
123	ERROR_INVALID_NAME

AIT_ExecCommand

機能

MS-DOS コマンドまたはシステムコマンドを実行します。

形式

```
bool AIT_ExecCommand  (
  string strCommandName     // MS-DOSコマンド
);
```

引数

strCommandName**（入力用）**

アプリケーションファイルの名前またはシステムコマンドを指定してください。

戻り値

この関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
2	ERROR_FILE_NOT_FOUND
3	ERROR_PATH_NOT_FOUND
5	ERROR_ACCESS_DENIED
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
21	ERROR_NOT_READY
87	ERROR_INVALID_PARAMETER
123	ERROR_INVALID_NAME

AIT_ExistWindow

機能

指定したウィンドウ名およびクラス名と一致するウィンドウがあるかどうかをチェックします。

形式

```
integer AIT_ExistWindow  (
  string strWndCaption,     // ウィンドウのキャプション
  string strClassName       // クラス名
  [,float fTimeOut]         // タイムアウト時間
);
```

引数

strWndCaption**（入力用）**

ウィンドウのキャプションを指定してください。

strClassName（入力用）

ウィンドウのクラス名を指定してください。

fTimeOut（入力用，省略可）

この関数がコントロールを見つけるために使用できる最大時間を秒単位で指定してください。省略した場合は，関数 AIT_SetDefaultWaitTimeout で設定した値が使用されます。

戻り値

ウィンドウが存在する場合，戻り値は 1 です。ウィンドウが存在しない場合，戻り値は 0 です。失敗した場合，戻り値は -1 です。-1 が返された場合は，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

AIT_Exit

機能

AIT ファイルの実行を終了します。

形式

```
AIT_Exit ();
```

引数

なし

戻り値

なし

AIT_FileClose

機能

ファイルハンドルを閉じます。

形式

```
bool AIT_FileClose  (
  integer nFileHandle     // ファイルハンドル
);
```

引数

nFileHandle（入力用）

AIT_FileOpen 関数を使って開いたファイルハンドルを指定してください。

戻り値

この関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
2	ERROR_FILE_NOT_FOUND
3	ERROR_PATH_NOT_FOUND
4	ERROR_TOO_MANY_OPEN_FILES
5	ERROR_ACCESS_DENIED
6	ERROR_INVALID_HANDLE
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY

AIT_FileCopy

機能

コピー元ファイルをコピー先ファイルにコピーします。コピー先に同じファイルがすでに存在する場合は，そのファイルが上書きされます。

形式

```
bool AIT_FileCopy  (
  string strSourceFileName,     // コピー元のファイル名
  string strTargetFileName      // コピー先のファイル名
);
```

引数

strSourceFileName（入力用）

コピー元のファイルの名前を指定してください。ワイルドカード（*）も使用できます。

strTargetFileName（入力用）

コピー先ファイルまたはディレクトリの名前を指定してください。ワイルドカードは使用できません。コピー先ディレクトリが存在しない場合は，ディレクトリが作成されます。

戻り値

この関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
2	ERROR_FILE_NOT_FOUND
3	ERROR_PATH_NOT_FOUND
5	ERROR_ACCESS_DENIED
87	ERROR_INVALID_PARAMETER
117	ERROR_INVALID_CATEGORY

AIT_FileDelete

機能

指定したファイルを削除します。

形式

```
bool AIT_FileDelete  (
  string strFileName     // ファイル名
);
```

引数

strFileName**（入力用）**

削除するファイルの名前を指定してください。ワイルドカード（*）も使用できます。

戻り値

この関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
2	ERROR_FILE_NOT_FOUND
3	ERROR_PATH_NOT_FOUND
5	ERROR_ACCESS_DENIED
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
15	ERROR_INVALID_DRIVE
19	ERROR_WRITE_PROTECT
21	ERROR_NOT_READY
53	ERROR_BAD_NETPATH
123	ERROR_INVALID_NAME
148	ERROR_PATH_BUSY
161	ERROR_BAD_PATHNAME
1005	ERROR_UNRECOGNIZED_VOLUME
1210	ERROR_INVALID_COMPUTERNAME
1214	ERROR_INVALID_NETNAME
1231	ERROR_NETWORK_UNREACHABLE

AIT_FileEOF

機能

ファイルポインタがファイルの終わりに到達したかどうかを確認します。

形式

```
integer AIT_FileEOF (
  integer nFileHandle     // ファイルハンドル
);
```

引数

nFileHandle（**入力用**）

ファイルハンドルを指定してください。

戻り値

EOF の場合，戻り値は 1 です。EOF ではない場合，戻り値は 0 です。失敗した場合，戻り値は -1 です。
-1 が返された場合は，AIT_GetLastError を使用して拡張エラーコードを取得できます。
AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
5	ERROR_ACCESS_DENIED
6	ERROR_INVALID_HANDLE
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
21	ERROR_NOT_READY
32	ERROR_SHARING_VIOLATION
33	ERROR_LOCK_VIOLATION
112	ERROR_DISK_FULL
148	ERROR_PATH_BUSY
1231	ERROR_NETWORK_UNREACHABLE
1392	ERROR_FILE_CORRUPT

AIT_FileExists

機能

指定した属性でファイルが存在するかどうかを確認します。

形式

```
integer AIT_FileExists  (
  string strFileName            // ファイル名
  [,integer nFileAttributes]    // ファイル属性
);
```

引数

strFileName（**入力用**）

検索するファイルの名前を指定してください。

nFileAttributes（**入力用，省略可**）

ファイルの属性を指定してください。これは，次の値のどれかにする必要があります。

値	意味
FILE_ATTRIBUTE_DIRECTORY	ファイルはディレクトリです。
FILE_ATTRIBUTE_HIDDEN	ファイルは隠しファイルです。
FILE_ATTRIBUTE_SYSTEM	ファイルは OS の一部，または OS 専用です。
FILE_ATTRIBUTE_ARCHIVE	ファイルはアーカイブファイルです。アプリケーションはこの属性を，ファイルのバックアップや削除のためのマークとして使います。
FILE_ATTRIBUTE_READONLY	ファイルは読み取り専用です。アプリケーションはファイルを読み取れますが，書き込みや削除はできません。

省略した場合は，ファイル属性とは無関係にファイルが検出されます。

戻り値

ファイルが存在する場合，戻り値は 1 です。ファイルが存在しない場合，戻り値は 0 です。失敗した場合，戻り値は -1 です。-1 が返された場合は，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
2	ERROR_FILE_NOT_FOUND
3	ERROR_PATH_NOT_FOUND
5	ERROR_ACCESS_DENIED
6	ERROR_INVALID_HANDLE
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
15	ERROR_INVALID_DRIVE
21	ERROR_NOT_READY
53	ERROR_BAD_NETPATH
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
123	ERROR_INVALID_NAME
148	ERROR_PATH_BUSY
161	ERROR_BAD_PATHNAME
998	ERROR_NOACCESS
1005	ERROR_UNRECOGNIZED_VOLUME
1210	ERROR_INVALID_COMPUTERNAME
1214	ERROR_INVALID_NETNAME
1231	ERROR_NETWORK_UNREACHABLE

AIT_FileGetLine

機能

指定したファイルから，データを読み取ります。

形式

```
bool AIT_FileGetLine  (
  integer nFileHandle,     // ファイルハンドル
  string strReadData       // ファイルから読み取るデータ
);
```

引数

nFileHandle**（入力用）**

ファイルハンドルを指定してください。

strReadData**（出力用）**

ファイルから読み取るデータを受け取る変数を指定してください。関数から制御が戻ると，この変数にデータが格納されます。

戻り値

この関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
5	ERROR_ACCESS_DENIED
6	ERROR_INVALID_HANDLE
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
21	ERROR_NOT_READY
32	ERROR_SHARING_VIOLATION
33	ERROR_LOCK_VIOLATION
38	ERROR_HANDLE_EOF
112	ERROR_DISK_FULL
148	ERROR_PATH_BUSY
1231	ERROR_NETWORK_UNREACHABLE
1392	ERROR_FILE_CORRUPT

AIT_FileGetPos

機能

ファイルポインタの現在位置を取得します。

形式

```
bool AIT_FileGetPos  (
  integer nFileHandle,     // ファイルハンドル
```

```
  integer nFilePos          // 現在のファイルポインタ位置
);
```

引数

nFileHandle（入力用）

ファイルハンドルを指定してください。

nFilePos（出力用）

現在のファイルポインタの位置を受け取る変数を指定してください。関数から制御が戻ると，この変数にポインタ位置が格納されます。

戻り値

この関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
5	ERROR_ACCESS_DENIED
6	ERROR_INVALID_HANDLE
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
21	ERROR_NOT_READY
32	ERROR_SHARING_VIOLATION
33	ERROR_LOCK_VIOLATION
38	ERROR_HANDLE_EOF
112	ERROR_DISK_FULL
148	ERROR_PATH_BUSY
1231	ERROR_NETWORK_UNREACHABLE
1392	ERROR_FILE_CORRUPT

AIT_FileOpen

機能

既存のファイルを開きます。または，指定したアクセスモードで新規ファイルを作成します。

形式

```
bool AIT_FileOpen  (
  string strFileName,      // ファイル名
  integer nAccessMode,     // アクセスモード
  integer nOperation,      // 作成方法
  integer nFileHandle      // ファイルハンドル
);
```

引数

strFileName（入力用）

作成するファイルまたは開くファイルの名前を指定してください。

nAccessMode（入力用）

ファイルへのアクセスモードを指定してください。これは，次の値のどれかまたは両方の組み合わせです。

値	意味
GENERIC_READ	ファイルへの読み取りアクセスを指定します。ファイルからのデータの読み取りとファイルポインタの移動が可能です。読み取り / 書き込みアクセスを指定するには，GENERIC_WRITE と組み合わせてください。
GENERIC_WRITE	ファイルへの書き込みアクセスを指定します。ファイルへのデータの書き込みとファイルポインタの移動が可能です。読み取り / 書き込みアクセスを指定するには，GENERIC_READ と組み合わせてください。

nOperation（入力用）

ファイルが存在する場合，または存在しない場合のファイルの扱い方を指定します。次の値のどれかにする必要があります。

値	意味
CREATE_NEW	新規ファイルを作成します。指定したファイルがすでに存在する場合，関数は失敗します。
CREATE_ALWAYS	新規ファイルを作成します。指定したファイルがすでに存在する場合，関数はそのファイルを上書きします。
OPEN_EXISTING	ファイルを開きます。ファイルが存在しない場合，関数は失敗します。
OPEN_ALWAYS	ファイルが存在する場合は，それを開きます。ファイルが存在しない場合は，そのファイルが作成されます。
TRUNCATE_EXISTING	ファイルを開きます。開かれたファイルは，サイズが 0 バイトになるように切り捨てられます。呼び出しプロセスで，少なくとも GENERIC_WRITE アクセスを指定してファイルを開く必要があります。ファイルが存在しない場合，関数は失敗します。

nFileHandle（出力用）

ファイルハンドルを受け取る変数を指定してください。関数から制御が戻ると，この変数にファイルハンドルが格納されます。

戻り値

この関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
2	ERROR_FILE_NOT_FOUND
3	ERROR_PATH_NOT_FOUND
4	ERROR_TOO_MANY_OPEN_FILES
5	ERROR_ACCESS_DENIED
6	ERROR_INVALID_HANDLE
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
15	ERROR_INVALID_DRIVE

拡張エラー番号	エラーコード
18	ERROR_NO_MORE_FILES
19	ERROR_WRITE_PROTECT
21	ERROR_NOT_READY
32	ERROR_SHARING_VIOLATION
33	ERROR_LOCK_VIOLATION
53	ERROR_BAD_NETPATH
80	ERROR_FILE_EXISTS
82	ERROR_CANNOT_MAKE
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
123	ERROR_INVALID_NAME
148	ERROR_PATH_BUSY
161	ERROR_BAD_PATHNAME
183	ERROR_ALREADY_EXISTS
206	ERROR_FILENAME_EXCED_RANGE
1005	ERROR_UNRECOGNIZED_VOLUME
1210	ERROR_INVALID_COMPUTERNAME
1214	ERROR_INVALID_NETNAME
1231	ERROR_NETWORK_UNREACHABLE
1392	ERROR_FILE_CORRUPT

注意事項

AIT_OpenFile が返したファイルハンドルを閉じるには，AIT_FileClose 関数を使用してください。

AIT_FilePutLine

機能

指定したファイルに，データを書き込みます。

形式

```
bool AIT_FilePutLine  (
  integer nFileHandle,      // ファイルハンドル
  string strWriteData       // ファイルに書き込むデータ
);
```

引数

nFileHandle**（入力用）**

ファイルハンドルを指定してください。

strWriteData**（入力用）**

ファイルに書き込むデータを指定してください。

戻り値

この関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
5	ERROR_ACCESS_DENIED
6	ERROR_INVALID_HANDLE
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
19	ERROR_WRITE_PROTECT
21	ERROR_NOT_READY
32	ERROR_SHARING_VIOLATION
33	ERROR_LOCK_VIOLATION
38	ERROR_HANDLE_EOF
112	ERROR_DISK_FULL
148	ERROR_PATH_BUSY
1231	ERROR_NETWORK_UNREACHABLE
1392	ERROR_FILE_CORRUPT

注意事項

- 0 バイトの書き込みを指定すると，null 書き込み操作の指定として解釈されます。
- AIT_FilePutLine は，現在のファイルポインタ位置にデータを書き込みます。ファイルポインタ位置は，書き込み操作後に更新されます。

AIT_FileRename

機能

ファイルまたはディレクトリの名前を変更します。

形式

```
bool AIT_FileRename (
  string strFileName,        // 現在のファイル名
  string strNewFileName      // 新しいファイル名
);
```

引数

strFileName**（入力用）**

名前を変更するファイルまたはディレクトリの名前を指定してください。

strNewFileName**（入力用）**

ファイルまたはディレクトリの新しい名前を指定してください。

戻り値

この関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
2	ERROR_FILE_NOT_FOUND
3	ERROR_PATH_NOT_FOUND
5	ERROR_ACCESS_DENIED
6	ERROR_INVALID_HANDLE
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
15	ERROR_INVALID_DRIVE
19	ERROR_WRITE_PROTECT
21	ERROR_NOT_READY
38	ERROR_HANDLE_EOF
53	ERROR_BAD_NETPATH
80	ERROR_FILE_EXISTS
82	ERROR_CANNOT_MAKE
87	ERROR_INVALID_PARAMETER
123	ERROR_INVALID_NAME
148	ERROR_PATH_BUSY
161	ERROR_BAD_PATHNAME
1005	ERROR_UNRECOGNIZED_VOLUME
1210	ERROR_INVALID_COMPUTERNAME
1214	ERROR_INVALID_NETNAME
1231	ERROR_NETWORK_UNREACHABLE

AIT_FileSetPos

機能

ファイルポインタを指定した位置に設定します。

形式

```
bool AIT_FileSetPos  (
  integer nFileHandle,      // ファイルハンドル
  integer nSetPos           // 設定後のファイルポインタの位置
);
```

引数

nFileHandle**（入力用）**

ファイルハンドルを指定してください。

nSetPos（入力用）

設定後のファイルポインタの位置を指定してください。

戻り値

この関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
5	ERROR_ACCESS_DENIED
6	ERROR_INVALID_HANDLE
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
19	ERROR_WRITE_PROTECT
21	ERROR_NOT_READY
32	ERROR_SHARING_VIOLATION
33	ERROR_LOCK_VIOLATION
38	ERROR_HANDLE_EOF
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
148	ERROR_PATH_BUSY
1231	ERROR_NETWORK_UNREACHABLE
1392	ERROR_FILE_CORRUPT

注意事項

- nFileHandle 引数の値で示すファイルポインタは，重複する読み取り操作や書き込み操作に使用しないでください。
- 設定後のファイルポインタの位置に 0 を指定してこの関数を実行すると，現在のファイルポインタ位置が保持されます。

AIT_FileSize

機能

ファイルのサイズを取得します。

形式

```
bool AIT_FileSize  (
  integer nFileHandle,     // ファイルハンドル
  integer nFileSize        // ファイルサイズ
);
```

引数

nFileHandle（入力用）

ファイルハンドルを指定してください。

nFileSize（出力用）

ファイルサイズを受け取る変数を指定してください。関数から制御が戻ると，この変数にファイルサイズが格納されます。

戻り値

この関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
2	ERROR_FILE_NOT_FOUND
5	ERROR_ACCESS_DENIED
6	ERROR_INVALID_HANDLE
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
21	ERROR_NOT_READY
32	ERROR_SHARING_VIOLATION
33	ERROR_LOCK_VIOLATION
38	ERROR_HANDLE_EOF
112	ERROR_DISK_FULL
148	ERROR_PATH_BUSY
1231	ERROR_NETWORK_UNREACHABLE
1392	ERROR_FILE_CORRUPT

注意事項

AIT_FileSize 関数は，圧縮されていないファイルサイズを取得します。

AIT_FindCloseFile

機能

AIT_FindFirstFile 関数によって返されたファイル検索ハンドルを閉じます。

形式

```
bool AIT_FindCloseFile  (
  integer nSearchHandle     // ファイル検索ハンドル
);
```

引数

nSearchHandle（入力用）

AIT_FindFirstFile 関数によって返されたファイルの検索ハンドルを指定してください。

戻り値

この関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が

返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
6	ERROR_INVALID_HANDLE
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
112	ERROR_DISK_FULL

注意事項

AIT_FindClose 関数の呼び出し後，nSearchHandle 引数で指定したハンドルを，AIT_FindNextFile 関数または AIT_FindCloseFile 関数への後続の呼び出しに使用することはできません。

AIT_FindFirstFile

機能

ファイル検索ハンドルを開いて，指定したファイル名と名前が一致する最初のファイル名を返します。

形式

```
bool AIT_FindFirstFile  (
  string strFileNamePattern,    // ファイル名
  string strFileName,           // 検索されたファイル名
  integer nSearchHandle         // ファイル検索ハンドル
);
```

引数

strFileNamePattern（入力用）

有効なディレクトリの名前，パスまたはファイル名を指定してください。ワイルドカード（*）も使用できます。文字列がワイルドカード，ピリオド，またはディレクトリ名で終わる場合，ユーザはルートまたはパス上のすべてのサブディレクトリへのアクセス権を持っている必要があります。

strFileName（出力用）

指定したファイル名と一致する，検索されたファイルの名前を受け取る変数を指定してください。関数から制御が戻ると，この変数に検索されたファイルの名前が格納されます。

nSearchHandle（出力用）

AIT_FindNextFile および AIT_FindCloseFile への後続の検索で使用するファイル検索ハンドルを受け取る変数を指定してください。関数から制御が戻ると，この変数にファイル検索ハンドルが格納されます。

戻り値

この関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
2	ERROR_FILE_NOT_FOUND
3	ERROR_PATH_NOT_FOUND

拡張エラー番号	エラーコード
5	ERROR_ACCESS_DENIED
6	ERROR_INVALID_HANDLE
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
18	ERROR_NO_MORE_FILES
21	ERROR_NOT_READY
112	ERROR_DISK_FULL
123	ERROR_INVALID_NAME
148	ERROR_PATH_BUSY
1005	ERROR_UNRECOGNIZED_VOLUME
1210	ERROR_INVALID_COMPUTERNAME
1214	ERROR_INVALID_NETNAME
1231	ERROR_NETWORK_UNREACHABLE

注意事項

- この関数は，ファイルを名前だけで検索します。属性での検索には使用できません。後続の「¥」の有無に関係なく，ルートディレクトリを AIT_FindFirstFile の strFileName 入力文字列として指定することはできません。
- AIT_FindFirstFile によって返されたファイル検索ハンドルを閉じるには，AIT_FindCloseFile を使用してください。

AIT_FindNextFile

機能

AIT_FindFirstFile 関数によって返された検索ハンドルで，次のファイルを検索します。

形式

```
bool AIT_FindNextFile  (
  integer nSearchHandle,     // ファイル検索ハンドル
  string strFileName         // 検索されたファイル名
);
```

引数

nSearchHandle**（入力用）**

AIT_FindFirstFile への前の呼び出しによって返されたファイル検索ハンドルを指定してください。

strFileName**（出力用）**

検索されたファイルの名前を受け取る変数を指定してください。関数から制御が戻ると，この変数に検索されたファイルの名前が格納されます。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す

可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
5	ERROR_ACCESS_DENIED
6	ERROR_INVALID_HANDLE
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
18	ERROR_NO_MORE_FILES
21	ERROR_NOT_READY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
148	ERROR_PATH_BUSY
1231	ERROR_NETWORK_UNREACHABLE

注意事項

この関数は，ファイルを名前だけで検索します。属性ベースの検索には使用できません。

AIT_FindSubStr

機能

指定した検索文字列を，文字列内の nStartPos で指定した位置から検索し，一致する最初の文字列の位置を返します。

形式

```
integer AIT_FindSubStr  (
  string strStrName,      // 文字列
  string strSearchStr     // 検索文字列
  [,integer nStartPos]    // 検索を始める文字の位置
);
```

引数

strStrName**（入力用）**

文字列を指定してください。

strSearchStr**（入力用）**

検索対象文字列を指定してください。

nStartPos**（入力用，省略可）**

文字列の検索を始める位置を指定してください。nStartPos の基準値は 0 で，0 の位置は文字列の最初の文字に対応します。省略した場合は，最初の文字から検索します。

戻り値

検索対象文字列の最初の文字につき基準値 0 のインデックスを返します。文字列が含まれない場合とヌル文字列が指定されている場合は -1 を返します。

AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあ

るエラーコードを次に示します。

拡張エラー番号	エラーコード
87	ERROR_INVALID_PARAMETER

AIT_FocusWindow

機能

指定したウィンドウ名とクラス名を持つウィンドウハンドルを取得して，フォーカスを設定します。

形式

```
integer AIT_FocusWindow  (
  string strWndCaption,     // ウィンドウのキャプション
  string strClassName       // クラス名
  [,float fTimeOut]         // タイムアウト時間
);
```

引数

strWndCaption**（入力用）**

ウィンドウのキャプションを指定してください。

strClassName**（入力用）**

ウィンドウのクラス名を指定してください。

fTimeOut**（入力用，省略可）**

この関数がコントロールを見つけるために使用できる最大時間を秒単位で指定してください。省略した場合は，AIT_SetDefaultWaitTimeout 関数の値が使用されます。

戻り値

関数が正常に処理された場合のウィンドウハンドルが戻り値となり，処理が失敗した場合の戻り値は 0 となります。関数が 0 を返した場合は，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

注意事項

strWndCaption に空の文字列を指定すると，空白のキャプションが付いたウィンドウを検索します。空白のキャプションが付いたウィンドウが複数ある場合は，最初に発見されたウィンドウにフォーカスを設定します。

AIT_GetCtrlText

機能

アクティブなウィンドウの特定のコントロールからテキストを取得します。

形式

```
bool AIT_GetCtrlText  (
  string strCaption,     // コントロールのキャプション
  integer nCtrlType,     // コントロールタイプ
  string strCtrlText     // コントロールのテキスト
  [,float fTimeOut]      // タイムアウト時間
);
bool AIT_GetCtrlText  (
  integer nCtrlID,      // コントロールID
  integer nCtrlType,    // コントロールタイプ
  string strCtrlText    // コントロールのテキスト
  [,float fTimeOut]     // タイムアウト時間
);
```

引数

strCaption（入力用）

コントロールのキャプションを指定してください。

nCtrlID（入力用）

コントロール ID を指定してください。

nCtrlType（入力用）

コントロールタイプを指定してください。これは，次の値のどれかにする必要があります。

値	意味
BUTTON_CTRL	コントロールタイプはコマンドボタンです。
CHECKBOX_CTRL	コントロールタイプはチェックボックスです。
OPTIONBUTTON_CTRL	コントロールタイプはオプションボタンです。
EDIT_CTRL	コントロールタイプはエディットボックスです。
STATIC_CTRL	コントロールタイプはスタティックテキストコントロールです。
COMBO_CTRL	コントロールタイプはコンボボックスです。
LISTBOX_CTRL	コントロールタイプはリストボックスです。
SPIN_CTRL	コントロールタイプはスピンコントロールです。
TREE_CTRL	コントロールタイプはツリーコントロールです。
LIST_CTRL	コントロールタイプはリストコントロールです。
DTPICKER_CTRL	コントロールタイプは日時ピッカーです。

strCtrlText（出力用）

コントロールのテキストを受け取る変数を指定してください。関数から制御が戻ると，この変数にテキストが格納されます。

fTimeOut（入力用，省略可）

この関数がコントロールを見つけるために使用できる最大時間を秒単位で指定してください。省略した場

合は，AIT_SetDefaultWaitTimeout 関数の値が使用されます。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
6	ERROR_INVALID_HANDLE
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

注意事項

- エディットボックスについては，エディットボックスの内容のテキストを取得します。スタティックテキストとボタンについては，コントロールのキャプションを取得します。そのほかのコントロールについては，その時点で選択されている項目をテキストとして取得します。
- コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連するラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_GetCtrlTextLen

機能

アクティブなウィンドウの特定のコントロールからテキストの長さを取得します。

形式

```
bool AIT_GetCtrlTextLen  (
  string strCaption,     // コントロールのキャプション
  integer nCtrlType,     // コントロールタイプ
  integer nTextLen       // コントロールテキストの長さ
  [,float fTimeOut]      // タイムアウト時間
);
bool AIT_GetCtrlTextLen  (
  integer nCtrlID,      // コントロールID
  integer nCtrlType,    // コントロールタイプ
  integer nTextLen      // コントロールテキストの長さ
  [,float fTimeOut]     // タイムアウト時間
);
```

引数

strCaption**（入力用）**

コントロールのキャプションを指定してください。

nCtrlID（入力用）

コントロール ID を指定してください。

nCtrlType（入力用）

コントロールタイプを指定してください。これは，次の値のどれかにする必要があります。

値	意味
BUTTON_CTRL	コントロールタイプはコマンドボタンです。
CHECKBOX_CTRL	コントロールタイプはチェックボックスです。
OPTIONBUTTON_CTRL	コントロールタイプはオプションボタンです。
EDIT_CTRL	コントロールタイプはエディットボックスです。
STATIC_CTRL	コントロールタイプはスタティックテキストです。
COMBO_CTRL	コントロールタイプはコンボボックスです。
LISTBOX_CTRL	コントロールタイプはリストボックスです。
SPIN_CTRL	コントロールタイプはスピンコントロールです。
TREE_CTRL	コントロールタイプはツリーコントロールです。
LIST_CTRL	コントロールタイプはリストコントロールです。
DTPICKER_CTRL	コントロールタイプは日時ピッカーです。

nTextLen（出力用）

コントロールテキストの長さを受け取る変数を指定してください。関数から制御が戻ると，この変数にテキストの長さが格納されます。

fTimeOut（入力用，省略可）

この関数がコントロールを見つけるために使用できる最大時間を秒単位で指定してください。省略した場合は，AIT_SetDefaultWaitTimeout 関数の値が使用されます。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
6	ERROR_INVALID_HANDLE
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

注意事項

コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連す

るラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_GetCurrentDirectory

機能

カレントディレクトリを取得します。

形式

```
bool AIT_GetCurrentDirectory  (
  string strDirName      // ディレクトリ名
);
```

引数

strDirName**（出力用）**

ディレクトリ名を受け取る変数を指定してください。関数から制御が戻ると，この変数にディレクトリ名が格納されます。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY

AIT_GetDate

機能

システムの日付を短い形式で取得します。

形式

```
string AIT_GetDate ();
```

引数

なし

戻り値

システムの日付を短い形式で文字列として返します。

AIT_GetDtPickerDate

機能

日時ピッカーから日付を取得します。

形式

```
bool AIT_GetDtPickerDate  (
  string strCaption,     // コントロールのキャプション
  string strOutDate      // コントロールの日付
  [,float fTimeOut]      // タイムアウト時間
);
bool AIT_GetDtPickerDate  (
  string strCaption,    // コントロールのキャプション
  integer nYear,        // 年
  integer nMonth,       // 月
  integer nDay          // 日
  [,float fTimeOut]     // タイムアウト時間
);
bool AIT_GetDtPickerDate  (
  integer nCtrlID,      // コントロールID
  string strOutDate     // コントロールの日付
  [,float fTimeOut]     // タイムアウト時間
);
bool AIT_GetDtPickerDate  (
  integer nCtrlID,     // コントロールID
  integer nYear,       // 年
  integer nMonth,      // 月
  integer nDay         // 日
  [,float fTimeOut]    // タイムアウト時間
);
```

引数

strCaption（入力用）

コントロールのキャプションを指定してください。

nCtrlID（入力用）

コントロール ID を指定してください。

strOutDate（出力用）

コントロールの日付の値を受け取る変数を設定してください。関数から制御が戻ると，この変数に日付が格納されます。日付の値は *YYYY/MM/DD* という形式になり，*YYYY* が年，*MM* が月，*DD* が日を表します。

nYear（出力用）

コントロールの年の値を受け取る変数を指定してください。関数から制御が戻ると，この変数に年が格納されます。

nMonth（出力用）

コントロールの月の値を受け取る変数を指定してください。関数から制御が戻ると，この変数に月が格納されます。

nDay（出力用）

コントロールの日の値を受け取る変数を指定してください。関数から制御が戻ると，この変数に日が格納されます。

fTimeOut（入力用，省略可）

この関数がコントロールを見つけるために使用できる最大時間を秒単位で指定してください。省略した場合は，AIT_SetDefaultWaitTimeout 関数の値が使用されます。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。
AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
6	ERROR_INVALID_HANDLE
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

注意事項

コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連するラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_GetDtPickerTime

機能

日時ピッカーの時間を取得します。

形式

```
bool AIT_GetDtPickerTime  (
  string strCaption,    // コントロールのキャプション
  string strOutTime     // コントロールの時間
  [,float fTimeOut]     // タイムアウト時間
);
bool AIT_GetDtPickerTime  (
  string strCaption,    // コントロールID
  integer nHour,        // 時間
  integer nMinute,      // 分
  integer nSecond       // 秒
  [,float fTimeOut]     // タイムアウト時間
);
bool AIT_GetDtPickerTime  (
  integer nCtrlID,      // コントロールID
  string strOutTime     // コントロールの時間
  [,float fTimeOut]     // タイムアウト時間
);
bool AIT_SetDtPickerTime  (
  integer nCtrlID,      // コントロールID
  integer nHour,        // 時間
  integer nMinute,      // 分
  integer nSecond       // 秒
  [,float fTimeOut]     // タイムアウト時間
);
```

引数

strCaption（入力用）

コントロールのキャプションを指定してください。

nCtrlID（入力用）

コントロール ID を指定してください。

strOutTime（出力用）

コントロールの時刻の値を受け取る変数を設定してください。関数から制御が戻ると，この変数に時間が格納されます。時間の値は *hh:mm:ss* という形式になり，*hh* が時間，*mm* が分，*ss* が秒を表します。

nHour（出力用）

コントロールの時間の値を受け取る変数を指定してください。関数から制御が戻ると，この変数に時間が格納されます。

nMinute（出力用）

コントロールの分の値を受け取る変数を指定してください。関数から制御が戻ると，この変数に分が格納されます。

nSecond（出力用）

コントロールの秒の値を受け取る変数を指定してください。関数から制御が戻ると，この変数に秒が格納されます。

fTimeOut（入力用，省略可）

この関数がコントロールを見つけるために使用できる最大時間を秒単位で指定してください。省略した場合は，AIT_SetDefaultWaitTimeout 関数の値が使用されます。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
6	ERROR_INVALID_HANDLE
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

注意事項

コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連するラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分

を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_GetEditCurrentLineIndex

機能

複数行のエディットボックスの，カレント行のインデックスを取得します。カレント行とは，入力位置が設定されている行を指します。

形式

```
bool AIT_GetEditCurrentLineIndex  (
  string strCaption,   // コントロールのキャプション
  integer nIndex       // カレント行のインデックス
  [,float fTimeOut]    // タイムアウト時間
);
bool AIT_GetEditCurrentLineIndex  (
  integer nCtrlID,     // コントロールID
  integer nIndex       // カレント行のインデックス
  [,float fTimeOut]    // タイムアウト時間
);
```

引数

strCaption**（入力用）**

コントロールのキャプションを指定してください。

nCtrlID**（入力用）**

コントロール ID を指定してください。

nIndex**（出力用）**

カレント行のインデックスを受け取る変数を指定してください。関数から制御が戻ると，この変数にカレント行のインデックスが格納されます。インデックスの基準値は 0 です。

fTimeOut**（入力用，省略可）**

この関数がコントロールを見つけるために使用できる最大時間を秒単位で指定してください。省略した場合は，AIT_SetDefaultWaitTimeout 関数の値が使用されます。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
6	ERROR_INVALID_HANDLE
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE

拡張エラー番号	エラーコード
1460	ERROR_TIMEOUT

注意事項

コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連するラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_GetEditFirstLineIndex

機能

複数行のエディットボックスに表示される最初の行，または単一行のエディットボックスに表示される最初の文字のインデックスを取得します。

形式

```
bool AIT_GetEditFirstLineIndex  (
  string strCaption,      // コントロールのキャプション
  integer nFirstVisible   // 最初の行または文字のインデックス
  [,float fTimeOut]       // タイムアウト時間
);
bool AIT_GetEditFirstLineIndex  (
  integer nCtrlID,        // コントロールID
  integer nFirstVisible   // 最初の行または文字のインデックス
  [,float fTimeOut]       // タイムアウト時間
);
```

引数

strCaption（入力用）

コントロールのキャプションを指定してください。

nCtrlID（入力用）

コントロール ID を指定してください。

nFirstVisible（出力用）

最初の行または最初の文字の，インデックスの値を受け取る変数を指定してください。関数から制御が戻ると，この変数にインデックスが格納されます。インデックスの基準値は 0 です。

fTimeOut（入力用，省略可）

この関数がコントロールを見つけるために使用できる最大時間を秒単位で指定してください。省略した場合は，AIT_SetDefaultWaitTimeout 関数の値が使用されます。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。
AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
6	ERROR_INVALID_HANDLE
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

注意事項

コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連するラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_GetEditTextLineLen

機能

アクティブなウィンドウの複数行のエディットボックスにある任意の行の長さを取得します。

形式

```
bool AIT_GetEditTextLineLen  (
  string strCaption,     // コントロールのキャプション
  integer nLineIndex,    // 行のインデックス
  integer nLineLength    // 行の長さ
  [,float fTimeOut]      // タイムアウト時間
);
bool AIT_GetEditTextLineLen  (
  integer nCtrlID,       // コントロールID
  integer nLineIndex,    // 行のインデックス
  integer nLineLength    // 行の長さ
  [,float fTimeOut]      // タイムアウト時間
);
```

引数

strCaption**（入力用）**

コントロールのキャプションを指定してください。

nCtrlID**（入力用）**

コントロール ID を指定してください。

nLineIndex**（入力用）**

複数行のエディットボックスのインデックスを指定してください。インデックスの基準値は 0 です。

nLineLength**（出力用）**

nLineIndex で指定した行のインデックスの長さを受け取る変数を指定してください。関数から制御が戻ると，この変数に長さが格納されます。

fTimeOut（入力用，省略可）

この関数がコントロールを見つけるために使用できる最大時間を秒単位で指定してください。省略した場合は，AIT_SetDefaultWaitTimeout 関数の値が使用されます。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
6	ERROR_INVALID_HANDLE
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

注意事項

コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連するラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_GetEnv

機能

指定した環境変数の内容を取得します。

形式

```
string AIT_GetEnv  (
  string strEnvVar      // 環境変数名
);
```

引数

strEnvVar（入力用）

環境変数名を指定してください。

戻り値

環境変数の内容を返します。

AIT_GetErrorText

機能

指定したエラーコードに対応するシステムエラーテキストを取得します。

形式

```
string AIT_GetErrorText  (
  integer nErrorCode     // エラーコード
);
```

引数

nErrorCode（入力用）

AIT_GetLastError 関数が返したエラーコードを指定してください。

戻り値

指定したエラーコードに対応するエラーメッセージを返します。

AIT_GetIndexText

機能

アクティブなウィンドウの特定のコントロールから，インデックスで指定した項目テキストを取得します。

形式

```
bool AIT_GetIndexText  (
  string strCaption,    // コントロールのキャプション
  integer nCtrlType,    // コントロールタイプ
  integer nIndex,       // インデックス
  string strItemText    // 項目テキスト
  [,float fTimeOut]     // タイムアウト時間
);
bool AIT_GetIndexText  (
  integer nCtrlID,      // コントロールID
  integer nCtrlType,    // コントロールタイプ
  integer nIndex,       // インデックス
  string strItemText    // 項目テキスト
  [,float fTimeOut]     // タイムアウト時間
);
```

引数

strCaption（入力用）

コントロールのキャプションを指定してください。

nCtrlID（入力用）

コントロール ID を指定してください。

nCtrlType（入力用）

コントロールタイプを指定してください。これは，次の値のどれかにする必要があります。

値	意味
COMBO_CTRL	コントロールタイプはコンボボックスです。

値	意味
LISTBOX_CTRL	コントロールタイプはリストボックスです。
TREE_CTRL	コントロールタイプはツリーコントロールです。
LIST_CTRL	コントロールタイプはリストコントロールです。

nIndex（入力用）

取得する項目テキストのインデックスを指定してください。インデックスの基準値は 0 です。

strItemText（出力用）

コントロール上で指定したインデックスの項目テキストを受け取る変数を指定してください。関数から制御が戻ると，この変数に項目テキストが格納されます。

fTimeOut（入力用，省略可）

この関数がコントロールを見つけるために使用できる最大時間を秒単位で指定してください。省略した場合は，AIT_SetDefaultWaitTimeout 関数の値が使用されます。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
6	ERROR_INVALID_HANDLE
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1413	ERROR_INVALID_INDEX
1460	ERROR_TIMEOUT

注意事項

コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連するラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_GetIndexTextLen

機能

アクティブなウィンドウの特定のコントロールから，インデックスで指定した項目テキストの長さを取得します。

形式

```
bool AIT_GetIndexTextLen  (
  string strCaption,     // コントロールのキャプション
  integer nCtrlType,     // コントロールタイプ
  integer nIndex,        // インデックス
  integer nTextLen       // テキストの長さ
  [,float fTimeOut]      // タイムアウト時間
);
bool AIT_GetIndexTextLen  (
  integer nCtrlID,       // コントロールID
  integer nCtrlType,     // コントロールタイプ
  integer nIndex,        // インデックス
  integer nTextLen       // テキストの長さ
  [,float fTimeOut]      // タイムアウト時間
);
```

引数

strCaption（入力用）

コントロールのキャプションを指定してください。

nCtrlID（入力用）

コントロール ID を指定してください。

nCtrlType（入力用）

コントロールタイプを指定してください。これは，次の値のどれかにする必要があります。

値	意味
COMBO_CTRL	コントロールタイプはコンボボックスです。
LISTBOX_CTRL	コントロールタイプはリストボックスです。
TREE_CTRL	コントロールタイプはツリーコントロールです。
LIST_CTRL	コントロールタイプはリストコントロールです。

nIndex（入力用）

取得する項目テキストのインデックスを指定してください。インデックスの基準値は 0 です。

nTextLen（出力用）

コントロール上で指定したインデックスの，項目テキストの長さを受け取る変数を指定してください。関数から制御が戻ると，この変数に項目テキストの長さが格納されます。

fTimeOut（入力用，省略可）

この関数がコントロールを見つけるために使用できる最大時間を秒単位で指定してください。省略した場合は，AIT_SetDefaultWaitTimeout 関数の値が使用されます。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。
AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
6	ERROR_INVALID_HANDLE
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

注意事項

コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連するラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_GetKeyState

機能

キー状態を取得します。

形式

```
int AIT_GetKeyState  (
  integer nVirtualKey    // 仮想キー
);
```

引数

nVirtualKey**（入力用）**

キー状態を取得する仮想キーコードを指定してください。

これは，次の値のどれかにする必要があります。

値	意味
NUMLOCK	［Num Lock］キー
SCROLLLOCK	［Scroll Lock］キー
CAPSLOCK	［Caps Lock］キー

戻り値

キー状態がオンの場合の戻り値は 1，オフの場合は 0 となります。処理が失敗した場合の戻り値は -1 となります。

関数が -1 を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。
AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
87	ERROR_INVALID_PARAMETER

AIT_GetLastError

機能

直前に実行した関数の詳細コードを取得します。

形式

```
integer AIT_GetLastError ();
```

引数

なし

戻り値

詳細コードを返します。この詳細コードはランタイムエラーのコードと同じです。

AIT_GetMenu

機能

ウィンドウのメニューハンドルを取得するために使用します。

形式

```
bool AIT_GetMenu  (
  integer nWndHandle,     // ウィンドウハンドル
  integer nMenu           // メニューハンドル
  [,float fTimeOut]       // タイムアウト時間
);
```

引数

nWndHandle**（入力用）**

ウィンドウハンドルを指定してください。

nMenu**（出力用）**

メニューハンドルを受け取る変数を指定してください。関数から制御が戻ると，この変数にハンドルが格納されます。

fTimeOut**（入力用，省略可）**

この関数がコントロールを見つけるために使用できる最大時間を秒単位で指定してください。省略した場合は，AIT_SetDefaultWaitTimeout 関数の値が使用されます。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY

拡張エラー番号	エラーコード
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

AIT_GetMenuIndex

機能

指定したメニュー項目のインデックスを取得します。

形式

```
bool AIT_GetMenuIndex  (
  integer nMenu,          // メニューハンドル
  string strMenuText,     // メニュー項目
  integer nIndex          // メニュー項目のインデックス
  [,float fTimeOut]       // タイムアウト時間
);
```

引数

nMenu（入力用）

メニューハンドルを指定してください。

strMenuText（入力用）

メニュー項目を指定してください。

nIndex（出力用）

メニュー項目のインデックスを受け取る変数を指定してください。関数から制御が戻ると，この変数にインデックスが格納されます。インデックスの基準値は 0 です。

fTimeOut（入力用，省略可）

この関数がコントロールを見つけるために使用できる最大時間を秒単位で指定してください。省略した場合は，AIT_SetDefaultWaitTimeout 関数の値が使用されます。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
112	ERROR_DISK_FULL
1401	ERROR_INVALID_MENU_HANDLE
1460	ERROR_TIMEOUT

AIT_GetMenuText

機能

指定したメニュー項目を取得します。

形式

```
bool AIT_GetMenuText  (
  integer nMenu,         // メニューハンドル
  integer nIndex,        // メニュー項目のインデックス
  string strMenuText     // メニュー項目
  [,float fTimeOut]      // タイムアウト時間
);
```

引数

nMenu（入力用）

メニューハンドルを指定してください。

nIndex（入力用）

メニュー項目のインデックスを指定してください。インデックスの基準値は 0 です。

strMenuText（出力用）

メニュー項目を受け取る変数を指定してください。関数から制御が戻ると，この変数にメニュー項目が格納されます。

fTimeOut（入力用，省略可）

この関数がコントロールを見つけるために使用できる最大時間を秒単位で指定してください。省略した場合は，AIT_SetDefaultWaitTimeout 関数の値が使用されます。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1401	ERROR_INVALID_MENU_HANDLE
1460	ERROR_TIMEOUT

AIT_GetOSType

機能

OS のメジャーバージョン，マイナーバージョンおよびプラットフォーム ID を取得します。

形式

```
bool AIT_GetOSType  (
  integer nMajorVersion,    // OSのメジャーバージョン
  integer nMinorVersion,    // OSのマイナーバージョン
  integer nPlatformID       // プラットフォームID
);
```

引数

nMajorVersion（出力用）

OS のメジャーバージョンを受け取る変数を指定してください。関数から制御が戻ると，この変数に次のどれかの値が格納されます。

値	意味
4	Windows 98
4	Windows Me
4	Windows NT 4.0
5	Windows 2000
5	Windows XP
5	Windows Server 2003
6	Windows Vista
6	Windows Server 2008
6	Windows 7
6	Windows Server 2012
6	Windows 8

nMinorVersion（出力用）

OS のマイナーバージョンを受け取る変数を指定してください。関数から制御が戻ると，この変数に次のどれかの値が格納されます。

値	意味
10	Windows 98
90	Windows Me
0	Windows NT 4.0
0	Windows 2000
1	Windows XP
2	Windows Server 2003
0	Windows Vista
0	Windows Server 2008
1	Windows Server 2008 R2
1	Windows 7
2	Windows Server 2012
2	Windows 8

nPlatformID（出力用）

OS のプラットフォーム ID を受け取る変数を指定してください。関数から制御が戻ると，この変数に次のどれかの値が格納されます。

値	意味
VER_PLATFORM_WIN32_WINDOWS	Windows Me または Windows 98
VER_PLATFORM_WIN32_NT	Windows 8，Windows Server 2012，Windows 7，Windows Server 2008，Windows Vista，Windows Server 2003，Windows XP，Windows 2000，または Windows NT 4.0

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
122	ERROR_INSUFFICIENT_BUFFER

注意事項

アプリケーションが OS のどのバージョン上で稼働しているのかを確認する場合は，望ましいバージョン番号よりも大きい番号か同じ番号を使用してください。そうすることによって，より新しいバージョンの OS でも同じテストが実施可能であることがわかります。

AIT_GetProfileFirstSection

機能

指定した INI ファイルのセクションから，最初のキーとキーの値を取得します。

形式

```
bool AIT_GetProfileFirstSection  (
  string strIniFileName,     // INIファイル名
  string strSectionName,     // INIファイルのセクション名
  string strValues           // セクションのデータ
);
```

引数

strIniFileName（入力用）

INI ファイル名を指定してください。

strSectionName（入力用）

INI ファイルのセクション名を指定してください。

strValues（出力用）

指定したセクションからキーとキーの値を受け取る変数を指定してください。関数から制御が戻ると，この変数にキーとキーの値が格納されます。キーとキーの値の組み合わせは< key = value >の形式です。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。
AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
2	ERROR_FILE_NOT_FOUND
3	ERROR_PATH_NOT_FOUND
5	ERROR_ACCESS_DENIED
6	ERROR_INVALID_HANDLE
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
15	ERROR_INVALID_DRIVE
21	ERROR_NOT_READY
38	ERROR_HANDLE_EOF
53	ERROR_BAD_NETPATH
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
123	ERROR_INVALID_NAME
148	ERROR_PATH_BUSY
161	ERROR_BAD_PATHNAME
998	ERROR_NOACCESS
1005	ERROR_UNRECOGNIZED_VOLUME
1169	ERROR_NO_MATCH
1210	ERROR_INVALID_COMPUTERNAME
1214	ERROR_INVALID_NETNAME
1231	ERROR_NETWORK_UNREACHABLE

AIT_GetProfileFirstSectionNames

機能

指定した INI ファイルの最初のセクション名を取得します。

形式

```
bool AIT_GetProfileFirstSectionNames  (
  string strIniFileName,     // INIファイル名
  string strSectionName      // セクション名
);
```

引数

strIniFileName**（入力用）**

INI ファイル名を指定してください。

strSectionName（出力用）

指定した INI ファイルから最初のセクション名を受け取る変数名を指定してください。関数から制御が戻ると，この変数にセクション名が格納されます。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
2	ERROR_FILE_NOT_FOUND
3	ERROR_PATH_NOT_FOUND
5	ERROR_ACCESS_DENIED
6	ERROR_INVALID_HANDLE
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
15	ERROR_INVALID_DRIVE
21	ERROR_NOT_READY
53	ERROR_BAD_NETPATH
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
123	ERROR_INVALID_NAME
148	ERROR_PATH_BUSY
161	ERROR_BAD_PATHNAME
998	ERROR_NOACCESS
1005	ERROR_UNRECOGNIZED_VOLUME
1169	ERROR_NO_MATCH
1210	ERROR_INVALID_COMPUTERNAME
1214	ERROR_INVALID_NETNAME
1231	ERROR_NETWORK_UNREACHABLE

AIT_GetProfileNextSection

機能

AIT_GetProfileFirstSection 関数で指定した INI ファイルのセクションから次のキーとキーの値を取得します。

形式

```
bool AIT_GetProfileNextSection  (
  string strValues       // セクションのデータ
);
```

引数

strValues（出力用）

AIT_GetProfileFirstSection で指定した，セクションの次のキーとキーの値を受け取る変数を指定してください。関数から制御が戻ると，この変数にキーとキーの値が格納されます。キーとキーの値の組み合わせは＜ key = value ＞の形式です。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。キーが存在しない，セクションが存在しない，または指定した INI ファイルが存在しない場合，関数は false を返します。

AIT_GetProfileNextSectionNames

機能

AIT_GetProfileFirstSectionNames 関数で指定した INI ファイルから次のセクション名を取得します。

形式

```
bool AIT_GetProfileNextSectionNames  (
  string strSectionName       // セクション名
);
```

引数

strSectionName（出力用）

AIT_GetProfileFirstSectionNames で指定した INI ファイルの次のセクション名を受け取る変数を指定してください。関数から制御が戻ると，この変数にセクション名が格納されます。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。キーが存在しない，セクションが存在しない，または指定した INI ファイルが存在しない場合，関数は false を返します。

AIT_GetProfileString

機能

指定した INI ファイルのセクションから，指定したキーの値を取得します。

形式

```
bool AIT_GetProfileString  (
  string strIniFileName,     // INIファイル名
  string strSectionName,     // セクション名
  string strKeyName,         // キー名
  string strValue            // キーの値
);
```

引数

strIniFileName（入力用）

INI ファイル名を指定してください。

strSectionName（入力用）

INI ファイルのセクション名を指定してください。

strKeyName（入力用）

セクション名に属するキー名を指定してください。

strValue（出力用）

キーの値を受け取る変数を指定してください。関数から制御が戻ると，この変数にキーの値が格納されます。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
2	ERROR_FILE_NOT_FOUND
3	ERROR_PATH_NOT_FOUND
5	ERROR_ACCESS_DENIED
6	ERROR_INVALID_HANDLE
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
15	ERROR_INVALID_DRIVE
21	ERROR_NOT_READY
38	ERROR_HANDLE_EOF
53	ERROR_BAD_NETPATH
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
123	ERROR_INVALID_NAME
148	ERROR_PATH_BUSY
161	ERROR_BAD_PATHNAME
998	ERROR_NOACCESS
1005	ERROR_UNRECOGNIZED_VOLUME
1169	ERROR_NO_MATCH
1210	ERROR_INVALID_COMPUTERNAME
1214	ERROR_INVALID_NETNAME
1231	ERROR_NETWORK_UNREACHABLE

AIT_GetSubMenu

機能

メニューのサブメニューハンドルを取得します。

形式

```
bool AIT_GetSubMenu  (
  integer nMenu,       // メニューハンドル
  integer nIndex,      // メニュー項目のインデックス
  integer nSubMenu     // サブメニューハンドル
  [,float fTimeOut]    // タイムアウト時間
);
```

引数

nMenu（入力用）

AIT_GetMenu 関数を呼び出して取得したメニューハンドルを指定してください。

nIndex（入力用）

メニュー項目のインデックスを指定してください。インデックスの基準値は 0 です。

nSubMenu（出力用）

サブメニューハンドルを受け取る変数を指定してください。関数から制御が戻ると，この変数にハンドルが格納されます。

fTimeOut（入力用，省略可）

この関数がコントロールを見つけるために使用できる最大時間を秒単位で指定してください。省略した場合は，AIT_SetDefaultWaitTimeout 関数の値が使用されます。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1401	ERROR_INVALID_MENU_HANDLE
1460	ERROR_TIMEOUT

AIT_GetSubStr

機能

文字列から指定した長さの文字列を返します。

形式

```
bool AIT_GetSubStr (
  string  strSubString,    // 抽出した文字列
  string  strStrName,      // 文字列
  integer nStartPos        // 抽出を始める文字の位置
  [,integer nLength]       // 文字数
);
```

引数

strSubString（出力用）

抽出した文字列を受け取る変数を指定してください。関数から制御が戻ると，この変数に文字列が格納されます。

strStrName（入力用）

文字列名を指定してください。

nStartPos（入力用）

抽出を始める文字の位置を指定してください。基準値は 0 で，0 の位置は文字列の最初の文字に対応します。

nLength（入力用，省略可）

抽出する文字数を指定してください。この引数が文字列の文字数を超えないかぎり，nStartPos から指定した文字数の文字列が抽出されます。省略した場合は，文字列の末尾の文字までの長さとなります。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
87	ERROR_INVALID_PARAMETER

AIT_GetTime

機能

システムの時間を取得します。

形式

```
string AIT_GetTime ();
```

引数

なし

戻り値

システムの時間を取得します。

AIT_GetWindowText

機能

指定したウィンドウのタイトルを取得します。

形式

```
bool AIT_GetWindowText  (
```

```
  integer nWndHandle,     // ウィンドウハンドル
  string strCaption       // コントロールのキャプション
);
```

引数

nWndHandle（入力用）

ウィンドウハンドルを指定してください。0 が指定されている場合は，アクティブなウィンドウのタイトルを取得します。

strCaption（出力用）

コントロールのキャプションを受け取る変数を指定してください。関数から制御が戻ると，この変数にキャプションが格納されます。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
6	ERROR_INVALID_HANDLE
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE

AIT_IMEGetConversionStatus

機能

現在の IME 変換ステータスを取得します。

形式

```
bool AIT_IMEGetConversionStatus  (
  integer nWndHandle,     // ウィンドウハンドル
  integer nConvMode,      // 変換モード
  integer nSentenceMode   // 文モード
);
```

引数

nWndHandle（入力用）

ステータスを取得するウィンドウハンドルを指定してください。

0 が指定されている場合は，入力フォーカスを持つウィンドウが使用されます。

nConvMode（出力用）

変換ステータスを受け取る変数を指定してください。関数から制御が戻ると，この変数に次の値の組み合わせが格納されます。

値	意味
IME_CMODE_CHARCODE	オンの場合，IME のモードは文字コード入力モードです。
IME_CMODE_EUDC	オンの場合，IME のモードは EUDC 変換モードです。
IME_CMODE_FULLSHAPE	オンの場合，IME のモードは全角モードです。オフの場合，半角モードです。
IME_CMODE_HANJACONVERT	オンの場合，IME のモードは HANJA 変換モードです。
IME_CMODE_KATAKANA	オンの場合，かたかなモードです。オフの場合，ひらがなモードです。
IME_CMODE_NATIVE	オンの場合，NATIVE モードです。オフの場合，ALPHANUMERIC モードです。
IME_CMODE_NOCONVERSION	オンの場合，IME は変換を実行しません。IME を閉じている状態と同じです。
IME_CMODE_ROMAN	オンの場合，IME のモードはローマ字入力モードです。
IME_CMODE_SOFTKBD	オンの場合，IME のモードはソフトキーボードモードです。
IME_CMODE_SYMBOL	オンの場合，IME のモードは SYMBOL 変換モードです。

nSentenceMode**（出力用）**

文字モードを受け取る変数を指定してください。関数から制御が戻ると，この変数に次の値の組み合わせが格納されます。

値	意味
IME_SMODE_AUTOMATIC	IME は自動モードで変換します。
IME_SMODE_NONE	センテンスについての情報はありません。
IME_SMODE_PHRASEPREDICT	IME はフレーズ情報を利用して次の文字を予測します（連文節）。
IME_SMODE_PLURALCLAUSE	IME は変換処理する際に複数の文節情報を使用します（複合語優先）。
IME_SMODE_SINGLECONVERT	IME はシングルモードで変換します。
IME_SMODE_CONVERSATION	IME は変換モードを使用します。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
87	ERROR_INVALID_PARAMETER
1400	ERROR_INVALID_WINDOW_HANDLE

AIT_IMEGetOpenStatus

機能

IME が開いているか閉じているかをチェックします。

形式

```
integer AIT_IMEGetOpenStatus  (
  [integer nWndHandle]     // ウィンドウハンドル
);
```

引数

nWndHandle（**入力用，省略可**）

ステータスを取得するウィンドウハンドルを指定してください。

省略した場合は，入力フォーカスを持つウィンドウが使用されます。

戻り値

IME が開いている場合の戻り値は 1，IME が閉じている場合の戻り値は 0 となり，処理が失敗した場合の戻り値は -1 となります。

関数が -1 を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。
AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
87	ERROR_INVALID_PARAMETER
1400	ERROR_INVALID_WINDOW_HANDLE

AIT_IMEGetProperty

機能

入力フォーカスウィンドウと結び付けられた IME のプロパティまたは機能を取得します。

形式

```
integer AIT_IMEGetProperty  (
  integer nPropertyInfo     // プロパティ情報
);
```

引数

nPropertyInfo（**入力用**）

取得するプロパティ情報を指定してください。

次の値のどれかに該当する必要があります。

値	意味
IGP_PROPERTY	プロパティ情報
IGP_CONVERSION	変換機能
IGP_SENTENCE	センテンスモード機能
IGP_UI	ユーザーインターフェース機能
IGP_SETCOMPSTR	コンポジション文字列機能
IGP_SELECT	選択継承の機能

戻り値

関数が正常に処理された場合の戻り値は nPropertyInfo の値に応じたプロパティまたは機能の値となり，そのほかの場合の戻り値は -1 となります。

nPropertyInfo が IGP_PROPERTY の場合戻り値は次の値の組み合わせになります。

値	意味
IME_PROP_AT_CARET	オンの場合，変換ウィンドウはカレットの位置にあります。オフの場合，変換ウィンドウはカレットの近くにあります。
IME_PROP_SPECIAL_UI	オンの場合，IME は標準的なユーザーインターフェースを持っていません。この場合は，アプリケーションで IME ウィンドウ内を描画しないでください。
IME_PROP_CANDLIST_START_FROM_1	オンの場合，候補リスト内の文字列は 1 から順に番号が付いています。オフの場合，文字列は 0 から順に番号が付いています。
IME_PROP_UNICODE	オンの場合，入力コンテキスト文字列には Unicode 文字が含まれます。オフの場合，文字列には 1 バイト文字と 2 バイト文字が含まれます。

nPropertyInfo が IGP_UI の場合，戻り値は次の値の組み合わせになります。

値	意味
UI_CAP_2700	テキストの印字方向として，0 または 2700 の値をサポートします。
UI_CAP_ROT90	テキストの印字方向として，0，900，1800，または 2700 の値をサポートします。
UI_CAP_ROTANY	任意の印字方向をサポートします。

nPropertyInfo が IGP_SETCOMPSTR の場合，戻り値は次の値の組み合わせになります。

値	意味
SCS_CAP_COMPSTR	IMESetCompositionString の SCS_SETSTR 値を使用してコンポジション文字列を作成できます。
SCS_CAP_MAKEREAD	IMESetCompositionString の SCS_SETSTR 値を使用すると，対応するコンポジション文字列から読み取り文字列を作成できます。

nPropertyInfo が IGP_SELECT の場合，戻り値は次の値の組み合わせになります。

値	意味
SELECT_CAP_CONVERSION	IME が新たに選択された場合，変換モードを継承します。
SELECT_CAP_SENTENCE	IME が新たに選択された場合，センテンスモードを継承します。

関数が -1 を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。

AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
87	ERROR_INVALID_PARAMETER

AIT_IMEGetStatusWindowPos

機能

ステータスウィンドウの位置を取得します。

形式

```
bool AIT_IMEGetStatusWindowPos  (
  integer nWndHandle,     // ウィンドウハンドル
  integer nX,             // X座標
  integer nY              // Y座標
);
```

引数

nWndHandle**（入力用）**

ステータスウィンドウの位置を取得するウィンドウハンドルを指定してください。

0 が指定されている場合は，入力フォーカスを持つウィンドウハンドルが使用されます。

nX**（出力用）**

ステータスウィンドウの X 座標を受け取る変数を指定してください。関数から制御が戻ると，この変数に X 座標が格納されます。

nY**（出力用）**

ステータスウィンドウの Y 座標を受け取る変数指定してください。関数から制御が戻ると，この変数に Y 座標が格納されます。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
87	ERROR_INVALID_PARAMETER
1400	ERROR_INVALID_WINDOW_HANDLE

AIT_IMESetConversionStatus

機能

現在の変換ステータスを設定します。

形式

```
bool AIT_IMESetConversionStatus  (
  integer nWndHandle,     // ウィンドウハンドル
  integer nConvMode,      // 変換モード
  integer nSentenceMode   // 文モード
);
```

引数

nWndHandle（入力用）

ステータスを設定するウィンドウハンドルを指定してください。

0 が指定されている場合は，入力フォーカスを持つウィンドウハンドルが使用されます。

nConvMode（入力用）

変換モードの組み合わせを指定してください。

ビット値については，「AIT_IMEGetConversionStatus」を参照してください。

nSentenceMode（入力用）

文モードを指定してください。

ビット値については，「AIT_IMEGetConversionStatus」を参照してください。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。
AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
87	ERROR_INVALID_PARAMETER
1400	ERROR_INVALID_WINDOW_HANDLE

AIT_IMESetOpenStatus

機能

IME を開いたり，閉じたりします。

形式

```
bool AIT_IMESetOpenStatus  (
  integer nWndHandle,     // ウィンドウハンドル
  bool bCondition         // 条件
);
```

引数

nWndHandle（入力用）

ステータスを設定するウィンドウハンドルを指定してください。

0 が指定されている場合は，入力フォーカスを持つウィンドウハンドルが使用されます。

bCondition（入力用）

IME を開くか閉じるかを指定してください。true の場合 IME を開き，false の場合 IME を閉じます。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
1400	ERROR_INVALID_WINDOW_HANDLE

AIT_IMESetStatusWindowPos

機能

ステータスウィンドウの位置を設定します。

形式

```
bool AIT_IMESetStatusWindowPos  (
  integer nWndHandle,     // ウィンドウハンドル
  integer nX,             // X座標
  integer nY              // Y座標
);
```

引数

nWndHandle**（入力用）**

位置を設定するウィンドウハンドルを指定してください。

0 が指定されている場合は，入力フォーカスされているウィンドウハンドルが使用されます。

nX**（入力用）**

ステータスウィンドウの X 座標を指定してください。

nY**（入力用）**

ステータスウィンドウの Y 座標を指定してください。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
87	ERROR_INVALID_PARAMETER
1400	ERROR_INVALID_WINDOW_HANDLE

AIT_IMESimulateHotKey

機能

指定した IME ホットキーをシミュレートします。

形式

```
bool AIT_IMESimulateHotKey  (
```

```
  integer nWndHandle,     // ウィンドウハンドル
  integer nHotKeyId       // ホットキー識別子
);
```

引数

nWndHandle（入力用）

ウィンドウハンドルを指定してください。

nHotKeyId（入力用）

IME ホットキーの識別子を指定してください。

次の値のどれかに該当する必要があります。

値	意味
IME_CHOTKEY_IME_NONIME_TOGGLE	このホットキーは中国語（簡体字）版で使用します。IME オペレーションと非 IME オペレーションとを切り替えます。
IME_CHOTKEY_SHAPE_TOGGLE	このホットキーは中国語（簡体字）版で使用します。IME のシェイプ変換モードを切り替えます。
IME_CHOTKEY_SYMBOL_TOGGLE	このホットキーは中国語（簡体字）版で使用します。IME のシェイプ変換モードを切り替えます。シンボルモードで中国語の句読法とシンボル（完全な全角文字）を入力できるようにするには，それらをキーボードに割り当てます。
IME_JHOTKEY_CLOSE_OPEN	このホットキーは日本語で使用し，IME を開いたり閉じたりします。
IME_KHOTKEY_ENGLISH	このホットキーは韓国語で使用し，英語へ切り替えます。
IME_KHOTKEY_SHAPE_TOGGLE	このホットキーは韓国語で使用し，IME のシェイプ変換モードを切り替えます。
IME_KHOTKEY_HANJACONVERT	このホットキーは韓国語で使用し，漢字（Hanja）変換へ切り替えます。
IME_THOTKEY_IME_NONIME_TOGGLE	このホットキーは中国語（繁体字）版で使用し，IME オペレーションと非 IME オペレーションを切り替えます。
IME_THOTKEY_SHAPE_TOGGLE	このホットキーは中国語（繁体字）版で使用し，IME のシェイプ変換モードを切り替えます。
IME_THOTKEY_SYMBOL_TOGGLE	このホットキーは中国語（繁体字）版で使用し，IME のシンボル変換モードを切り替えます。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。
AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
87	ERROR_INVALID_PARAMETER
1400	ERROR_INVALID_WINDOW_HANDLE

AIT_InitLog

機能

AIT_LogMessage 関数が使用する RecDFile.log ファイルを初期化します。AIT_LogMessage 関数を実行する前には必ずこの関数を実行してください。RecDFile.log ファイルは，次のレジストリキー値で指定した LOG ディレクトリのパスに存在します。

- OS が 32 ビット版の場合
  `HKEY_LOCAL_MACHINE¥SOFTWARE¥HITACHI¥NETM/DM/P¥PathName`
- OS が 64 ビット版の場合
  `HKEY_LOCAL_MACHINE¥SOFTWARE¥Wow6432Node¥Hitachi¥NETM/DM/P¥PathName`

形式

```
bool AIT_InitLog  (
  string strMessage     // メッセージ文字列
);
```

引数

strMessage**（入力用）**

ログファイルに書き込む文字列メッセージを指定してください。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
3	ERROR_PATH_NOT_FOUND
4	ERROR_TOO_MANY_OPEN_FILES
5	ERROR_ACCESS_DENIED
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
19	ERROR_WRITE_PROTECT
32	ERROR_SHARING_VIOLATION
33	ERROR_LOCK_VIOLATION
53	ERROR_BAD_NETPATH
82	ERROR_CANNOT_MAKE
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
123	ERROR_INVALID_NAME
148	ERROR_PATH_BUSY
161	ERROR_BAD_PATHNAME
183	ERROR_ALREADY_EXISTS
206	ERROR_FILENAME_EXCED_RANGE
1005	ERROR_UNRECOGNIZED_VOLUME

拡張エラー番号	エラーコード
1210	ERROR_INVALID_COMPUTERNAME
1214	ERROR_INVALID_NETNAME
1231	ERROR_NETWORK_UNREACHABLE
1392	ERROR_FILE_CORRUPT

注意事項

- この関数を実行すると，以前の RecDFile.log は Rec1File.log に変更され，五つまで履歴として保存されます。
- この関数はメッセージとともに現在の日付と時間を記録します。

AIT_IsEmpty

機能

入力した文字列が空か，空でないかを確認します。

形式

```
bool AIT_IsEmpty (
  string strStrName     // 文字列名
);
```

引数

strStrName**（入力用）**

文字列名を指定してください。

戻り値

文字列が空の場合 true を返し，そのほかの場合は false を返します。

AIT_LogMessage

機能

AIT_InitLog 関数によって開かれた RecDFile.log ファイルにメッセージを記録します。

形式

```
bool AIT_LogMessage  (
  string strMessage     // メッセージ文字列
);
```

引数

strMessage**（入力用）**

ログファイルに書き込む文字列メッセージを指定してください。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す

可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
3	ERROR_PATH_NOT_FOUND
4	ERROR_TOO_MANY_OPEN_FILES
5	ERROR_ACCESS_DENIED
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
19	ERROR_WRITE_PROTECT
32	ERROR_SHARING_VIOLATION
33	ERROR_LOCK_VIOLATION
53	ERROR_BAD_NETPATH
82	ERROR_CANNOT_MAKE
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
123	ERROR_INVALID_NAME
148	ERROR_PATH_BUSY
161	ERROR_BAD_PATHNAME
183	ERROR_ALREADY_EXISTS
206	ERROR_FILENAME_EXCED_RANGE
1005	ERROR_UNRECOGNIZED_VOLUME
1210	ERROR_INVALID_COMPUTERNAME
1214	ERROR_INVALID_NETNAME
1231	ERROR_NETWORK_UNREACHABLE
1392	ERROR_FILE_CORRUPT

注意事項

この関数はメッセージとともに現在の日付と時間を記録します。

AIT_MenuItemClick

機能

指定したメニュー項目をクリックします。

形式

```
bool AIT_MenuItemClick  (
  integer nWndHandle,    // ウィンドウハンドル
  integer nMenu,         // メニューハンドル
  integer nIndex         // メニュー項目のインデックス
  [,float fTimeOut]      // タイムアウト時間
);
```

引数

nWndHandle**（入力用）**

ウィンドウハンドルを指定してください。

nMenu（入力用）

メニューハンドルを指定してください。

nIndex（入力用）

メニュー項目のインデックスを指定してください。インデックスの基準値は 0 です。

fTimeOut（入力用，省略可）

この関数がコントロールを見つけるために使用できる最大時間を秒単位で指定してください。省略した場合は，AIT_SetDefaultWaitTimeout 関数の設定値が使用されます。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1401	ERROR_INVALID_MENU_HANDLE
1460	ERROR_TIMEOUT

AIT_MessageBox

機能

指定したメッセージをダイアログボックスに表示し，ユーザがどれかのボタンをクリックするまで待機し，ユーザが選択したボタンを示す値を返します。

形式

```
integer AIT_MessageBox  (
  string strMessage,       // メッセージ
  string strTitle          // タイトル
  [,integer nIconType]     // アイコンタイプ
  [,integer nMsgBoxType]   // メッセージボックスタイプ
);
```

引数

strMessage（入力用）

メッセージボックスに表示するメッセージを指定してください。

strTitle（入力用）

メッセージボックスのタイトルを指定してください。

nIconType（入力用，省略可）

表示するアイコンタイプを指定してください。次の値のどれかに該当する必要があります。

値	意味
MB_ICONEXCLAMATION	メッセージボックスに「!」（感嘆符）のアイコンが表示されます。
MB_ICONINFORMATION	メッセージボックスに，丸の中に「i」があるアイコンが表示されます。
MB_ICONQUESTION	メッセージボックスに「?」（疑問符）のアイコンが表示されます。
MB_ICONSTOP	メッセージボックスに停止標識のアイコンが表示されます。

省略した場合は，MB_ICONEXCLAMATION がアイコンタイプになります。

nMsgBoxType（入力用，省略可）

メッセージボックスタイプを指定してください。次の値のどれかに該当する必要があります。

値	意味
MB_ABORTRETRYIGNORE	［中止］，［再試行］，および［無視］ボタンがあるメッセージボックス
MB_OK	［OK］ボタンだけがあるメッセージボックス
MB_OKCANCEL	［OK］および［キャンセル］ボタンがあるメッセージボックス
MB_RETRYCANCEL	［再試行］および［キャンセル］ボタンがあるメッセージボックス
MB_YESNO	［はい］および［いいえ］ボタンがあるメッセージボックス
MB_YESNOCANCEL	［はい］，［いいえ］，および［キャンセル］ボタンがあるメッセージボックス

省略した場合は，MB_OK がメッセージボックスタイプに指定されます。

戻り値

ユーザが選択したボタンを表す値を返します。次のどれかの値になります。

値	意味
IDABORT	［中止］ボタンが選択されました。
IDCANCEL	［キャンセル］ボタンが選択されました。
IDIGNORE	［無視］ボタンが選択されました。
IDNO	［いいえ］ボタンが選択されました。
IDOK	［OK］ボタンが選択されました。
IDRETRY	［再試行］ボタンが選択されました。
IDYES	［はい］ボタンが選択されました。

上記以外の値が返った場合は，AIT_GetLastError を使用して拡張エラーコードを取得できます。
AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
87	ERROR_INVALID_PARAMETER

AIT_MinWnd

機能

指定したウィンドウを最小化し，次の最上位のウィンドウがアクティブになります。

形式

```
bool AIT_MinWnd  (
  integer nWndHandle     // ウィンドウハンドル
);
bool AIT_MinWnd  (
  string strCaption,     // コントロールのキャプション
  string strClassName    // クラス名
);
```

引数

nWndHandle**（入力用）**

ウィンドウハンドルを指定してください。

strCaption**（入力用）**

コントロールのキャプションを指定してください。

strClassName**（入力用）**

ウィンドウのクラス名を指定してください。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
5	ERROR_ACCESS_DENIED
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE

AIT_MouseClick

機能

指定した座標でマウスをクリックします。

形式

```
bool AIT_MouseClick  (
  integer nMouseButton,     // マウスボタン
  integer nX,               // X座標
  integer nY                // Y座標
);
```

引数

nMouseButton（入力用）

クリックするマウスボタンを指定してください。これは，次の値のどれかにする必要があります。

値	意味
LBUTTON	左マウスボタン
MBUTTOM	中央マウスボタン
RBUTTON	右マウスボタン

nX（入力用）

クリックする位置の X 座標を指定してください。

nY（入力用）

クリックする位置の Y 座標を指定してください。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL

AIT_MouseDblClk

機能

指定した座標でマウスをダブルクリックします。

形式

```
bool AIT_MouseDblClk  (
  integer nX,             // X座標
  integer nY,             // Y座標
  integer nButton,        // マウスボタン
  integer nKeyState       // キー状態
);
```

引数

nX（入力用）

ダブルクリックする位置の X 座標を指定してください。

nY（入力用）

ダブルクリックする位置の Y 座標を指定してください。

nButton（入力用）

ダブルクリックするマウスボタンを指定してください。これは，次の値のどれかにする必要があります。

値	意味
LBUTTON	左マウスボタン
MBUTTON	中央マウスボタン
RBUTTON	右マウスボタン

nKeyState（入力用）

キー状態を指定してください。これは，次の値のどれかにする必要があります。

値	意味
SHIFT_ON	［Shift］キーはオン状態
ALT_ON	［Alt］キーはオン状態
CTRL_ON	［Ctrl］キーはオン状態
SHIFT_OFF	［Shift］キーはオフ状態
ALT_OFF	［Alt］キーはオフ状態
CTRL_OFF	［Ctrl］キーはオフ状態

戻り値

この関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。
AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL

AIT_MouseDown

機能

指定した座標でマウスのボタンを押します。

形式

```
bool AIT_MouseDown  (
  integer nX,               // X座標
  integer nY,               // Y座標
  integer nButton,          // マウスボタン
```

```
  integer nKeyState         // キー状態
);
```

引数

nX（入力用）

マウスボタンを押す位置の X 座標を指定してください。

nY（入力用）

マウスボタンを押す位置の Y 座標を指定してください。

nButton（入力用）

マウスで押すボタンを指定してください。これは，次の値のどれかにする必要があります。

値	意味
LBUTTON	左マウスボタン
MBUTTON	中央マウスボタン
RBUTTON	右マウスボタン

nKeyState（入力用）

キー状態を指定してください。これは，次の値のどれかにする必要があります。

値	意味
SHIFT_ON	［Shift］キーはオン状態
ALT_ON	［Alt］キーはオン状態
CTRL_ON	［Ctrl］キーはオン状態
SHIFT_OFF	［Shift］キーはオフ状態
ALT_OFF	［Alt］キーはオフ状態
CTRL_OFF	［Ctrl］キーはオフ状態

戻り値

この関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL

AIT_MouseDragDrop

機能

指定した開始位置から終了位置までをドラッグアンドドロップします。

形式

```
bool AIT_MouseDragDrop  (
  integer nXStartPos,       // 開始X座標
  integer nYStartPos,       // 開始Y座標
  integer nXEndPos,         // 終了X座標
  integer nYEndPos,         // 終了Y座標
  integer nButton,          // マウスボタン
  integer nKeyState         // キー状態
);
```

引数

nXStartPos（入力用）

ドラッグを開始する位置の X 座標を指定してください。

nYStartPos（入力用）

ドラッグを開始する位置の Y 座標を指定してください。

nXEndPos（入力用）

ドロップする位置の X 座標を指定してください。

nYEndPos（入力用）

ドロップする位置の Y 座標を指定してください。

nButton（入力用）

ドラッグアンドドロップするマウスボタンを指定してください。これは，次の値のどれかにする必要があります。

値	意味
LBUTTON	左マウスボタン
MBUTTON	中央マウスボタン
RBUTTON	右マウスボタン

nKeyState（入力用）

キー状態を指定してください。これは，次の値のどれかにする必要があります。

値	意味
SHIFT_ON	［Shift］キーはオン状態
ALT_ON	［Alt］キーはオン状態
CTRL_ON	［Ctrl］キーはオン状態
SHIFT_OFF	［Shift］キーはオフ状態
ALT_OFF	［Alt］キーはオフ状態

値	意味
CTRL_OFF	［Ctrl］キーはオフ状態

戻り値

この関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL

AIT_MouseMoveTo

機能

指定した座標へマウスを移動します。

形式

```
bool AIT_MouseMoveTo  (
  integer nX,              // X座標
  integer nY,              // Y座標
  integer nButton,         // マウスボタン
  integer nKeyState        // キー状態
);
```

引数

nX（入力用）

マウスを移動する位置の X 座標を指定してください。

nY（入力用）

マウスを移動する位置の Y 座標を指定してください。

nButton（入力用）

移動中にクリックするマウスボタンを指定してください。これは，次の値のどれかにする必要があります。

値	意味
LBUTTON	左マウスボタン
MBUTTON	中央マウスボタン
RBUTTON	右マウスボタン

nKeyState（入力用）

キー状態を指定してください。これは，次の値のどれかにする必要があります。

値	意味
SHIFT_ON	［Shift］キーはオン状態
ALT_ON	［Alt］キーはオン状態
CTRL_ON	［Ctrl］キーはオン状態
SHIFT_OFF	［Shift］キーはオフ状態
ALT_OFF	［Alt］キーはオフ状態
CTRL_OFF	［Ctrl］キーはオフ状態

戻り値

この関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL

AIT_MouseUp

機能

指定した座標でマウスのボタンを離します。

形式

```
bool AIT_MouseUp  (
  integer nX,           // X座標
  integer nY,           // Y座標
  integer nButton,      // マウスボタン
  integer nKeyState     // キー状態
);
```

引数

nX（入力用）

マウスボタンを離す位置の X 座標を指定してください。

nY（入力用）

マウスボタンを離す位置の Y 座標を指定してください。

nButton（入力用）

マウスで離すボタンを指定してください。これは，次の値のどれかにする必要があります。

値	意味
LBUTTON	左マウスボタン

値	意味
MBUTTON	中央マウスボタン
RBUTTON	右マウスボタン

nKeyState（入力用）

キー状態を指定してください。これは，次の値のどれかにする必要があります。

値	意味
SHIFT_ON	［Shift］キーはオン状態
ALT_ON	［Alt］キーはオン状態
CTRL_ON	［Ctrl］キーはオン状態
SHIFT_OFF	［Shift］キーはオフ状態
ALT_OFF	［Alt］キーはオフ状態
CTRL_OFF	［Ctrl］キーはオフ状態

戻り値

この関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL

AIT_PlayKey

機能

ユーザがキー入力した場合と同じように，アクティブなウィンドウへキーボード入力情報を送信します。

形式

```
bool AIT_PlayKey  (
  string strKeys     // 文字列
);
```

引数

strKeys（入力用）

キーやキーの組み合わせ文字列，または文字列を指定してください。次のどれかを組み合わせて指定できます。

- a から z までのすべての大文字と小文字
- 0 から 9 までのすべての数字

- 次の半角記号
  「~」「!」「@」「#」「$」「%」「^」「&」「*」「(」「)」「_」「+」「|」「?」「>」「<」「"」「}」「{」「[」「]」「'」「;」「/」「.」「,」「`」「-」「=」「¥」

例

AIT_PlayKey("ABC") は，ABC という文字列を打つ処理をシミュレートします。

次に示すのは指定が可能な特殊キーです。

指定可能な特殊キー						
{ALT}	{ALT_LOCK}	{ALT_UNLOCK}	{ALT+SHIFT}	{APPS}	{BACKSPACE}	{CTRL}
{CTRL+ALT}	{CTRL+SHIFT}	{CTRL+ALT+SHIFT}	{CTRL_LOCK}	{CAPSLOCK}	{CAPSLOCK_ON}	{CAPSLOCK_OFF}
{CTRL_UNLOCK}	{DELETE}	{DOWN}	{END}	{ENTER}	{ESC}	{F1}
{F2}	{F3}	{F4}	{F5}	{F6}	{F7}	{F8}
{F9}	{F10}	{F11}	{F12}	{HOME}	{INS}	{KANA_ON}
{KANA_OFF}	{KANJI_ON}	{KANJI_OFF}	{LEFT}	{LWIN}	{LWIN_LOCK}	{LWIN_UNLOCK}
{NUMLOCK_ON}	{NUMLOCK_OFF}	{NUMLOCK}	{NUMPAD0}	{NUMPAD1}	{NUMPAD2}	{NUMPAD3}
{NUMPAD4}	{NUMPAD5}	{NUMPAD6}	{NUMPAD7}	{NUMPAD8}	{NUMPAD9}	{PAUSE}
{PGDOWN}	{PGUP}	{PRNSCR}	{PROCESS}	{RIGHT}	{RWIN}	{RWIN_LOCK}
{RWIN_UNLOCK}	{SCROLLLOCK}	{SCROLLLOCK_ON}	{SCROLLLOCK_OFF}	{SHIFT}	{SHIFT_LOCK}	{SHIFT_UNLOCK}
{SPACE}	{TAB}	{UP}	—	—	—	—

（凡例）—：値なし

例

AIT_PlayKey("{TAB}") は［Tab］キーを押す処理をシミュレートします。

［Shift］,［Ctrl］, または［Alt］キーと組み合わせたキーを指定する場合は，通常のキーテキストの前に次のキー操作コードを付け加えてください。

キー	キー操作コード
Shift	+
Ctrl	^
Alt	%

例

AIT_PlayKey("%(N)") は［Alt］+［N］キーを押すことを表します。
同じ順番でキーを繰り返す場合は，次のように表記してください。
{REPEAT n}<char to repeat>{END_REPEAT}

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。
AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
87	ERROR_INVALID_PARAMETER

AIT_PostMessage

機能

指定したウィンドウを作成したスレッドに関連づけられているメッセージキューにメッセージをポストします。対応するスレッドがメッセージを処理するのを待つことなく制御を返します。

形式

```
bool AIT_PostMessage  (
  integer nWndHandle,     // ウィンドウハンドル
  integer nMessage,       // メッセージ
  integer nWParam,        // メッセージの最初の引数
  integer nLParam         // メッセージの2番目の引数
);
```

引数

nWndHandle（入力用）

ポスト先のウィンドウハンドルを指定してください。

次の値は特別な意味を持ちます。

値	意味
HWND_BROADCAST	システム内のすべてのトップレベルウィンドウへメッセージをポストします。無効になっている所有されていないウィンドウ，不可視の所有されていないウィンドウ，オーバーラップされた（手前にほかのウィンドウがあって覆い隠されている）ウィンドウ，ポップアップウィンドウもポスト先になります。子ウィンドウへはメッセージをポストしません。

nMessage（入力用）

ポストするメッセージを指定してください。

nWParam（入力用）

メッセージ特有の追加情報を指定してください。

nLParam（入力用）

メッセージ特有の追加情報を指定してください。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。
AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER

AIT_RegCloseKey

機能

指定したレジストリキーのハンドルを閉じます。

形式

```
bool AIT_RegCloseKey(
  integer nHKeyHandle     // キーハンドル
);
```

引数

nHKeyHandle**（入力用）**

すでに開かれているレジストリキーを指定してください。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
6	ERROR_INVALID_HANDLE
87	ERROR_INVALID_PARAMETER
1009	ERROR_BADDB
1010	ERROR_BADKEY
1015	ERROR_REGISTRY_CORRUPT
1016	ERROR_REGISTRY_IO_FAILED
1019	ERROR_NO_LOG_SPACE

AIT_RegCreateKey

機能

指定したレジストリキーを作成します。すでに存在している場合はそのキーを開きます。

形式

```
bool AIT_RegCreateKey(
  integer nHKeyHandle,       // キーハンドル
  string strRegKeyName,      // 作成するキー名
  integer nOutputHkeyHandle  // 出力用のキーハンドル
);
```

引数

nHKeyHandle（入力用）

すでに開かれているレジストリキーのハンドルまたは次のどれかを指定してください。

- HKEY_CLASSES_ROOT
- HKEY_CURRENT_CONFIG
- HKEY_CURRENT_USER
- HKEY_LOCAL_MACHINE
- HKEY_USERS

strRegKeyName（入力用）

作成または開くレジストリキー名を指定してください。

nOutputHkeyHandle（出力用）

レジストリキーのハンドルを受け取る変数を指定してください。関数から制御が戻ると，この変数にハンドルが格納されます。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
6	ERROR_INVALID_HANDLE
87	ERROR_INVALID_PARAMETER
1009	ERROR_BADDB
1010	ERROR_BADKEY
1015	ERROR_REGISTRY_CORRUPT
1016	ERROR_REGISTRY_IO_FAILED
1019	ERROR_NO_LOG_SPACE

注意事項

● 終了時に AIT_RegCloseKey を呼び出して nOutputHkeyHandle を閉じてください。

● レジストリキーを作成するには，作成位置へ書き込みできる権限が必要です。

● すでに存在しているレジストリキーを開く場合，そのレジストリキーに書き込みできる権限がないときは，読み込み権限で開きます。

AIT_RegDeleteKey

機能

指定したレジストリのサブキーを削除します。

形式

```
bool AIT_RegDeleteKey(
```

```
  integer nHKeyHandle,      // キーハンドル
  string strRegKeyName      // 削除するキー名
);
```

引数

nHKeyHandle（入力用）

すでに開かれたレジストリキーのハンドルを指定してください。

strRegKeyName（入力用）

削除するレジストリのサブキー名を指定してください。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
2	ERROR_FILE_NOT_FOUND
5	ERROR_ACCESS_DENIED
6	ERROR_INVALID_HANDLE
87	ERROR_INVALID_PARAMETER
1009	ERROR_BADDB
1010	ERROR_BADKEY
1015	ERROR_REGISTRY_CORRUPT
1016	ERROR_REGISTRY_IO_FAILED
1019	ERROR_NO_LOG_SPACE

注意事項

クライアントの OS が Windows NT の場合，指定したキーにサブキーがあるときは，指定したキーを削除できません。あらかじめサブキーを削除しておいてください。Windows Me または Windows 98 の場合は，サブキーを含めて指定したキーを削除できます。

AIT_RegDeleteValue

機能

指定したレジストリ値を削除します。

形式

```
bool AIT_RegDeleteValue(
  integer nHKeyHandle,      // キーハンドル
  string strRegValueName    // レジストリ値名
);
```

引数

nHKeyHandle（入力用）

すでに開かれたレジストリキーのハンドルを指定してください。

strRegValueName（入力用）

削除するレジストリ値名を指定してください。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
2	ERROR_FILE_NOT_FOUND
6	ERROR_INVALID_HANDLE
87	ERROR_INVALID_PARAMETER
1009	ERROR_BADDB
1010	ERROR_BADKEY
1015	ERROR_REGISTRY_CORRUPT
1016	ERROR_REGISTRY_IO_FAILED
1019	ERROR_NO_LOG_SPACE

AIT_RegGetDWORDValue

機能

DWORD のデータ型のレジストリ値を取得します。

形式

```
bool AIT_RegGetDWORDValue(
  integer nHKeyHandle,      // キーハンドル
  string strRegKeyName,     // レジストリサブキー名
  string strRegValueName,   // レジストリ値名
  integer nRegValueData     // レジストリ値データ
);
```

引数

nHKeyHandle（入力用）

すでに開かれているレジストリキーのハンドルまたは次のどれかを指定してください。

- HKEY_CLASSES_ROOT
- HKEY_CURRENT_CONFIG
- HKEY_CURRENT_USER
- HKEY_LOCAL_MACHINE
- HKEY_USERS

strRegKeyName（入力用）

レジストリのサブキー名を指定してください。

strRegValueName（入力用）

取得するレジストリ値名を指定してください。

nRegValueData（出力用）

DWORD のデータ型のレジストリ値データを受け取る変数を指定してください。関数から制御が戻ると，この変数にレジストリ値データが格納されます。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
2	ERROR_FILE_NOT_FOUND
6	ERROR_INVALID_HANDLE
87	ERROR_INVALID_PARAMETER
234	ERROR_MORE_DATA
1009	ERROR_BADDB
1010	ERROR_BADKEY
1011	ERROR_CANTOPEN
1012	ERROR_CANTREAD
1015	ERROR_REGISTRY_CORRUPT
1016	ERROR_REGISTRY_IO_FAILED
1018	ERROR_KEY_DELETED
1019	ERROR_NO_LOG_SPACE
1069	ERROR_NO_MATCH

AIT_RegGetStringValue

機能

文字列のデータ型（REG_SZ，REG_EXPAND_SZ，REG_MULTI_SZ）のレジストリ値を取得します。

形式

```
bool AIT_RegGetStringValue(
  integer nHKeyHandle,      // キーハンドル
  string strRegKeyName,     // レジストリサブキー名
  string strRegValueName,   // レジストリ値名
  string strRegValueData    // レジストリ値データ
);
```

引数

nHKeyHandle（入力用）

すでに開かれているレジストリキーのハンドルまたは次のどれかを指定してください。

- HKEY_CLASSES_ROOT
- HKEY_CURRENT_CONFIG
- HKEY_CURRENT_USER
- HKEY_LOCAL_MACHINE
- HKEY_USERS

strRegKeyName（入力用）

レジストリのサブキー名を指定してください。

strRegValueName（入力用）

取得するレジストリ値名を指定してください。

strRegValueData（出力用）

文字列のデータ型のレジストリ値データを受け取る変数を指定してください。関数から制御が戻ると，この変数のレジストリ値データが格納されます。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
2	ERROR_FILE_NOT_FOUND
6	ERROR_INVALID_HANDLE
87	ERROR_INVALID_PARAMETER
234	ERROR_MORE_DATA
1009	ERROR_BADDB
1010	ERROR_BADKEY
1011	ERROR_CANTOPEN
1012	ERROR_CANTREAD
1015	ERROR_REGISTRY_CORRUPT
1016	ERROR_REGISTRY_IO_FAILED
1018	ERROR_KEY_DELETED
1019	ERROR_NO_LOG_SPACE
1169	ERROR_NO_MATCH

AIT_RegisterWindowMessage

機能

システム上で固有となる新たなウィンドウメッセージを定義します。メッセージの送信またはポストをする際に，このメッセージ値を利用できます。

通常この関数は二つの強調アプリケーション間の通信に使うメッセージを登録する目的で使います。

形式

```
integer AIT_RegisterWindowMessage  (
  string strMessageString     // メッセージ文字列
);
```

引数

strMessageString**（入力用）**

登録するメッセージを指定してください。

戻り値

メッセージが正常に登録された場合，戻り値は 49152 ～ 65535 のメッセージ値（メッセージ識別子）になります。

関数の処理が失敗した場合の戻り値は 0 となります。AIT_GetLastError を使用して拡張エラーコードを取得できます。

AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER

AIT_RegKeyExists

機能

指定したレジストリキーが存在するかを確認します。

形式

```
integer AIT_RegKeyExists(
  integer nHKeyHandle,       // キーハンドル
  string strRegKeyName       // レジストリサブキー名
);
```

引数

nHKeyHandle**（入力用）**

すでに開かれているレジストリキーのハンドルを指定してください。

strRegKeyName**（入力用）**

存在を確認するレジストリのサブキー名を指定してください。

戻り値

このキーが存在する場合の戻り値は 1，このキーが存在しない場合は 0 となり，処理が失敗した場合の戻り値は -1 となります。

関数が -1 を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。
AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
2	ERROR_FILE_NOT_FOUND
6	ERROR_INVALID_HANDLE
87	ERROR_INVALID_PARAMETER
1009	ERROR_BADDB
1010	ERROR_BADKEY
1011	ERROR_CANTOPEN
1015	ERROR_REGISTRY_CORRUPT
1016	ERROR_REGISTRY_IO_FAILED
1019	ERROR_NO_LOG_SPACE

AIT_RegOpenKey

機能

指定したレジストリキーを開きます。

形式

```
bool AIT_RegOpenKey(
  integer nHKeyHandle,        // キーハンドル
  string strRegKeyName,       // 開かれるキー名
  integer nOutputHkeyHandle  // 出力用のキーハンドル
);
```

引数

nHKeyHandle**（入力用）**

すでに開かれているレジストリキーのハンドルまたは次のどれかを指定してください。

- HKEY_CLASSES_ROOT
- HKEY_CURRENT_CONFIG
- HKEY_CURRENT_USER
- HKEY_LOCAL_MACHINE
- HKEY_USERS

strRegKeyName**（入力用）**

開くレジストリのサブキー名を指定してください。

nOutputHkeyHandle**（出力用）**

レジストリキーのハンドルを受け取る変数を指定してください。関数から制御が戻ると，この変数にハンドルが格納されます。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。
AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
2	ERROR_FILE_NOT_FOUND
6	ERROR_INVALID_HANDLE
87	ERROR_INVALID_PARAMETER
1009	ERROR_BADDB
1010	ERROR_BADKEY
1011	ERROR_CANTOPEN
1015	ERROR_REGISTRY_CORRUPT
1016	ERROR_REGISTRY_IO_FAILED
1018	ERROR_KEY_DELETED
1019	ERROR_NO_LOG_SPACE

注意事項

- 終了時に AIT_RegCloseKey を呼び出して nOutputHkeyHandle を閉じてください。
- レジストリキーを開く場合，そのレジストリキーに書き込みできる権限がないときは，読み込み権限で開きます。

AIT_RegSetDWORDValue

機能

DWORD のデータ型のレジストリ値にデータを設定します。

形式

```
bool AIT_RegSetDWORDValue(
  integer nHKeyHandle,      // キーハンドル
  string strRegKeyName,     // レジストリサブキー名
  string strRegValueName,   // レジストリ値名
  integer nRegValueData     // レジストリ値データ
);
```

引数

nHKeyHandle**（入力用）**

すでに開かれているレジストリキーのハンドルまたは次のどれかを指定してください。

- HKEY_CLASSES_ROOT
- HKEY_CURRENT_CONFIG
- HKEY_CURRENT_USER
- HKEY_LOCAL_MACHINE
- HKEY_USERS

strRegKeyName（入力用）

レジストリのサブキー名を指定してください。

strRegValueName（入力用）

設定するレジストリ値名を指定してください。

nRegValueData（入力用）

DWORD のデータ型のレジストリ値に設定するデータを指定してください。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
6	ERROR_INVALID_HANDLE
87	ERROR_INVALID_PARAMETER
234	ERROR_MORE_DATA
1009	ERROR_BADDB
1010	ERROR_BADKEY
1011	ERROR_CANTOPEN
1013	ERROR_CANTWRITE
1015	ERROR_REGISTRY_CORRUPT
1016	ERROR_REGISTRY_IO_FAILED
1018	ERROR_KEY_DELETED
1019	ERROR_NO_LOG_SPACE
1169	ERROR_NO_MATCH

AIT_RegSetStringValue

機能

文字列のデータ型（REG_SZ，REG_EXPAND_SZ，REG_MULTI_SZ）のレジストリ値にデータを設定します。

形式

```
bool AIT_RegSetStringValue(
  integer nHKeyHandle,      // キーハンドル
  string strRegKeyName,     // レジストリサブキー名
  string strRegValueName,   // レジストリ値名
  string strRegValueData    // レジストリ値データ
);
```

引数

nHKeyHandle（入力用）

すでに開かれているレジストリキーのハンドルまたは次のどれかを指定してください。

- HKEY_CLASSES_ROOT
- HKEY_CURRENT_CONFIG
- HKEY_CURRENT_USER
- HKEY_LOCAL_MACHINE
- HKEY_USERS

strRegKeyName（入力用）

レジストリのサブキー名を指定してください。

strRegValueName（入力用）

設定するレジストリ値名を指定してください。

strRegValueData（入力用）

文字列のデータ型のレジストリ値に設定するデータを指定してください。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
6	ERROR_INVALID_HANDLE
87	ERROR_INVALID_PARAMETER
234	ERROR_MORE_DATA
1009	ERROR_BADDB
1010	ERROR_BADKEY
1011	ERROR_CANTOPEN
1013	ERROR_CANTWRITE
1015	ERROR_REGISTRY_CORRUPT
1016	ERROR_REGISTRY_IO_FAILED
1018	ERROR_KEY_DELETED
1019	ERROR_NO_LOG_SPACE
1169	ERROR_NO_MATCH

注意事項

- 新規にレジストリを作成する場合，文字列のデータ型は REG_SZ になります。データ型は変更できません。
- 既存のレジストリ値を更新する場合，API を実行してもデータ型は変更されません。

AIT_RegValueExists

機能

指定したレジストリ値が存在するかを確認します。

形式

```
integer AIT_RegValueExists(
  integer nHKeyHandle,      // キーハンドル
  string strRegValueName    // レジストリ値名
);
```

引数

nHKeyHandle（入力用）

すでに開かれているレジストリキーのハンドルを指定してください。

strRegValueName（入力用）

存在を確認するレジストリ値名を指定してください。

戻り値

この値が存在する場合の戻り値は 1，この値が存在しない場合の戻り値は 0 となり，処理が失敗した場合の戻り値は -1 となります。

関数が -1 を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。
AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
2	ERROR_FILE_NOT_FOUND
6	ERROR_INVALID_HANDLE
87	ERROR_INVALID_PARAMETER
236	ERROR_MORE_DATA
1009	ERROR_BADDB
1010	ERROR_BADKEY
1011	ERROR_CANTOPEN
1015	ERROR_REGISTRY_CORRUPT
1016	ERROR_REGISTRY_IO_FAILED
1019	ERROR_NO_LOG_SPACE

AIT_SelectIPAddressField

機能

アクティブなウィンドウの IP アドレスコントロール内のテキストを選択します。

形式

```
bool AIT_SelectIPAddressField  (
  string strCaption,        // コントロールのキャプション
  integer nFieldIndex,      //フィールドのインデックス
  integer nStartSel,        // テキストの開始位置
  integer nEndSel           // テキストの数
  [,float fTimeOut]         // タイムアウト時間
);
bool AIT_SelectIPAddressField  (
  integer nCtrlID,          // コントロールID
  integer nFieldIndex,      // フィールドのインデックス
  integer nStartSel,        // テキストの開始位置
```

```
  integer nEndSel           // テキストの数
  [,float fTimeOut]         // タイムアウト時間
);
```

引数

strCaption（入力用）

コントロールのキャプションを指定してください。

nCtrlID（入力用）

コントロール ID を指定してください。

nFieldIndex（入力用）

IP アドレスコントロール内のフィールドのインデックスを指定してください。インデックスの基準値は 0 です。

nStartSel（入力用）

テキストを選択する開始文字位置を指定してください。nStartSel の基準値は 0 で 0 の位置はテキストの最初の文字に対応します。

nEndSel（入力用）

選択するテキストの数を指定してください。

fTimeOut（入力用，省略可）

この関数がコントロールを見つけるために使用できる最大時間を秒単位で指定してください。省略した場合は，関数 AIT_SetDefaultWaitTimeout で設定した値が使用されます。

戻り値

この関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

注意事項

- 開始値が終了値より大きくてもかまいません。二つの値の小さい方が選択テキストの最初の文字位置を指定します。大きい方の値は選択テキストを超えた最初の文字位置を指定します。
- 開始値は選択テキストの固定点となり，終了値は変動する終点になります。
- 開始が 0 で終点が -1 の場合，エディットコントロール内のすべてのテキストが選択されます。開始が -1 の場合，アクティブな選択は解除されます。

● コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連するラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_SelectListItem

機能

アクティブなウィンドウの特定のコントロールで指定した項目を選択します。

形式

```
bool AIT_SelectListItem  (
  string strCaption,    // コントロールのキャプション
  integer nCtrlType,    // コントロールタイプ
  string strItemText    // 選択される項目テキスト
  [,float fTimeOut]     // タイムアウト時間
);
bool AIT_SelectListItem  (
  integer nCtrlID,      // コントロールID
  integer nCtrlType,    // コントロールタイプ
  string strItemText    // 選択される項目テキスト
  [,float fTimeOut]     // タイムアウト時間
);
```

引数

strCaption**（入力用）**

コントロールのキャプションを指定してください。

nCtrlID**（入力用）**

コントロール ID を指定してください。

nCtrlType**（入力用）**

コントロールタイプを指定してください。次の値のどれかに該当する必要があります。

値	意味
COMBO_CTRL	コントロールタイプはコンボボックスです。
LISTBOX_CTRL	コントロールタイプはリストボックスです。
LIST_CTRL	コントロールタイプはリストコントロールです。

strItemText**（入力用）**

特定のコントロール上で選択するテキストを指定してください。

fTimeOut**（入力用，省略可）**

この関数がコントロールを見つけるために使用できる最大時間を秒単位で指定してください。省略した場合は，AIT_SetDefaultWaitTimeout 関数の設定値が使用されます。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。

AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
6	ERROR_INVALID_HANDLE
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

注意事項

- AIT_SelectListItem 関数は複数選択可能なコントロールでは使用しないでください。
- コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連するラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_SelectMultipleListItem

機能

アクティブなウィンドウの特定の複数選択コントロールで指定した項目を選択します。

形式

```
bool AIT_SelectMultipleListItem  (
  string strCaption,     // コントロールのキャプション
  integer nCtrlType,     // コントロールタイプ
  string strItemText     // 選択される項目テキスト
  [,float fTimeOut]      // タイムアウト時間
);
bool AIT_SelectMultipleListItem  (
  integer nCtrlID,       // コントロールID
  integer nCtrlType,     // コントロールタイプ
  string strItemText     // 選択される項目テキスト
  [,float fTimeOut]      // タイムアウト時間
);
```

引数

strCaption**（入力用）**

コントロールのキャプションを指定してください。

nCtrlID**（入力用）**

コントロール ID を指定してください。

nCtrlType**（入力用）**

コントロールタイプを指定してください。次の値のどれかに該当する必要があります。

値	意味
LISTBOX_CTRL	コントロールタイプはリストボックスです。

値	意味
LIST_CTRL	コントロールタイプはリストコントロールです。

strItemText（入力用）

特定の複数選択コントロールで選択するテキストを指定してください。複数指定する場合はコンマで区切った値で指定できます。

fTimeOut（入力用，省略可）

この関数がコントロールを見つけるために使用できる最大時間を秒単位で指定してください。省略した場合は，AIT_SetDefaultWaitTimeout 関数の設定値が使用されます。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
6	ERROR_INVALID_HANDLE
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

注意事項

- AIT_SelectMultipleListItem 関数は複数選択可能なコントロールだけで使用してください。
- コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連するラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_SelectText

機能

アクティブなウィンドウのエディットコントロールでテキストを選択します。

形式

```
bool AIT_SelectText  (
  string strCaption,   // コントロールのキャプション
  integer nStartPos,   // コントロールの開始位置
  integer nEndPos      // コントロールの終了位置
  [,float fTimeOut]    // タイムアウト時間
);
bool AIT_SelectText  (
  integer nCtrlID,     // コントロールID
  integer nStartPos,   // コントロールの開始位置
```

```
  integer nEndPos       // コントロールの終了位置
  [,float fTimeOut]     // タイムアウト時間
);
```

引数

strCaption（入力用）

コントロールのキャプションを指定してください。

nCtrlID（入力用）

コントロール ID を指定してください。

nStartPos（入力用）

テキストを選択する開始文字位置を指定してください。nStartPos の基準値は 0 で，0 の位置はテキストの最初の文字に対応します。

nEndPos（入力用）

テキストを選択する終了文字位置を指定してください。

fTimeOut（入力用，省略可）

この関数がコントロールを見つけるために使用できる最大時間を秒単位で指定してください。省略した場合は，関数 AIT_SetDefaultWaitTimeout で設定した値が使用されます。

戻り値

この関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

注意事項

- 開始値が終了値より大きくてもかまいません。二つの値の小さい方が選択テキストの最初の文字位置を指定します。大きい方の値は選択テキストを超えた最初の文字位置を指定します。
- 開始値は選択テキストの固定点となり，終了値は変動する終点になります。
- 開始が 0 で終点が -1 の場合，エディットコントロール内のすべてのテキストが選択されます。開始が -1 の場合，アクティブな選択は解除されます。
- コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連するラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_SetActWnd

機能

指定したウィンドウをアクティブにします。

形式

```
bool AIT_SetActWnd  (
  integer nWndHandle     // ウィンドウハンドル
);
```

引数

nWndHandle（入力用）

ウィンドウのハンドルを指定してください。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false になります。関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE

AIT_SetCheck

機能

アクティブなウィンドウの特定のコントロールをオン，オフおよび，不確定状態にします。

形式

```
bool AIT_SetCheck  (
  string strCaption,     // コントロールのキャプション
  integer nCtrlType      // コントロールタイプ
  [,integer nCondition]  // チェックタイプ
  [,float fTimeOut]      // タイムアウト時間
);
bool AIT_SetCheck  (
  integer nCtrlID,       // コントロールID
  integer nCtrlType      // コントロールタイプ
  [,integer nCondition]  // チェック状態のタイプ
  [,float fTimeOut]      // タイムアウト時間
);
```

引数

strCaption（入力用）

コントロールのキャプションを指定してください。

nCtrlID（入力用）

コントロールの ID を指定してください。

nCtrlType（入力用）

コントロールタイプを指定してください。これは，次の値のどれかにする必要があります。

値	意味
CHECKBOX_CTRL	コントロールタイプがチェックボックスです。
OPTIONBUTTON_CTRL	コントロールタイプがオプションボタンです。

nCondition（入力用，省略可）

チェック状態のタイプを指定してください。これは，次の値のどれかにする必要があります。

値	意味
0	オフにした状態
1	オンにした状態
2	不確定またはグレー表示された状態

fTimeOut（入力用，省略可）

この関数がコントロールを見つけるために使用できる最大時間を秒単位で指定してください。省略した場合は，関数 AIT_SetDefaultWaitTimeout で設定した値が使用されます。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false になります。関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

注意事項

コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連するラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_SetComboEditSelText

機能

アクティブなウィンドウのコンボボックスでテキスト選択を設定します。

形式

```
bool AIT_SetComboEditSelText  (
  string strCaption,    // コントロールのキャプション
  integer nStartSel,    // コントロールの開始位置
  integer nEndSel       // コントロールの終了位置
  [,float fTimeOut]     // タイムアウト時間
);
bool AIT_SetComboEditSelText  (
  integer nCtrlID,      // コントロールのキャプション
  integer nStartSel,    // コントロールの開始位置
  integer nEndSel       // コントロールの終了位置
  [,float fTimeOut]     // タイムアウト時間
);
```

引数

strCaption（入力用）

コントロールのキャプションを指定してください。

nCtrlID（入力用）

コントロール ID を指定してください。

nStartSel（入力用）

エディットコントロール内のテキストを選択する開始位置を指定してください。nStartSel の基準値は 0 です。

nEndSel（入力用）

エディットコントロール内のテキストを選択する終了位置を指定してください。

fTimeOut（入力用，省略可）

この関数がコントロールを見つけるために使用できる最大時間を秒単位で指定してください。省略した場合は，関数 AIT_SetDefaultWaitTimeout で設定した値が使用されます。

戻り値

この関数が正常に処理された場合の戻り値は true，そのほかの場合は false となります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE

拡張エラー番号	エラーコード
1460	ERROR_TIMEOUT

注意事項

- 開始値が終了値より大きくてもかまいません。二つの値の小さい方が選択テキストの最初の文字位置を指定します。大きい方の値は，選択テキストを超えた最初の文字位置を指定します。
- 開始値は選択テキストの固定点となり，終了値は変動する終点になります。ユーザが［Shift］キーを使用して選択テキストのサイズを調整すると，開始が 0 で終点が -1 の場合，エディットコントロール内のすべてのテキストが選択されます。開始が -1 の場合，アクティブな選択は解除されます。
- コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連するラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_SetCurrentDirectory

機能

現在のディレクトリを，カレントディレクトリに変更します。

形式

```
bool AIT_SetCurrentDirectory  (
  string strDirName    // ディレクトリ名
);
```

引数

strDirName**（入力用）**

ディレクトリ名を指定してください。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false になります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。
AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
2	ERROR_FILE_NOT_FOUND
3	ERROR_PATH_NOT_FOUND
5	ERROR_ACCESS_DENIED
21	ERROR_NOT_READY
53	ERROR_BAD_NETPATH
123	ERROR_INVALID_NAME
161	ERROR_BAD_PATHNAME
1005	ERROR_UNRECOGNIZED_VOLUME
1210	ERROR_INVALID_COMPUTERNAME
1214	ERROR_INVALID_NETNAME

AIT_SetDefaultWaitTimeout

機能

コントロール関連の API で使うデフォルトのタイムアウト時間を設定します。

形式

```
AIT_SetDefaultWaitTimeout  (
  float fTimeOut      // タイムアウト値（秒）
);
```

引数

fTimeOut**（入力用）**

デフォルトのタイムアウト値を秒単位で指定してください。この関数を省略した場合はタイムアウト時間が 5 秒となります。

戻り値

なし

AIT_SetDtPickerDate

機能

日時ピッカーに日付を設定します。

形式

```
bool AIT_SetDtPickerDate  (
  string strCaption,     // コントロールのキャプション
  integer nYear,         // 年
  integer nMonth,        // 月
  integer nDay           // 日
  [,float fTimeOut]      // タイムアウト時間
);
bool AIT_SetDtPickerDate  (
  string strCaption,     // コントロールのキャプション
  string strInputDate    // 日付
  [,float fTimeOut]      // タイムアウト時間
);
bool AIT_SetDtPickerDate  (
  integer nCtrlID,       // コントロールID
  integer nYear,         // 年
  integer nMonth,        // 月
  integer nDay           // 日
  [,float fTimeOut]      // タイムアウト時間
);
bool AIT_SetDtPickerDate  (
  integer nCtrlID,        // コントロールID
  string  strInputDate    // 日付
  [,float fTimeOut]       // タイムアウト時間
);
```

引数

strCaption**（入力用）**

コントロールのキャプションを指定してください。

nCtrlID（入力用）

コントロール ID を指定してください。

nYear（入力用）

コントロールに設定する年を指定してください。

nMonth（入力用）

コントロールに設定する月を指定してください。

nDay（入力用）

コントロールに設定する日を指定してください。

strInputDate

[in] コントロールに設定する日付を指定してください。日付は *YYYY/MM/DD* 形式で，*YYYY* が年，*MM* が月，そして *DD* が日を示すように指定してください。

fTimeOut（入力用，省略可）

この関数がコントロールを見つけるために使用できる最大時間を秒単位で指定してください。省略した場合は，関数 AIT_SetDefaultWaitTimeout で設定した値が使用されます。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false になります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
6	ERROR_INVALID_HANDLE
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

注意事項

コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連するラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_SetDtPickerTime

機能

日時ピッカーに時刻を設定します。

形式

```
bool AIT_SetDtPickerTime  (
  string strCaption,     // コントロールのキャプション
  integer nHour,         // 時間
  integer nMinute,       // 分
  integer nSecond        // 秒
  [,float fTimeOut]      // タイムアウト時間
);
bool AIT_SetDtPickerTime  (
  string strCaption,     // コントロールのキャプション
  string strInputTime    // 時刻
  [,float fTimeOut]      // タイムアウト時間
);
bool AIT_SetDtPickerTime  (
  integer nCtrlID,       // コントロールID
  integer nHour,         // 時間
  integer nMinute,       // 分
  integer nSecond        // 秒
  integer [nTimeOut]     // タイムアウト時間
);
bool AIT_SetDtPickerTime  (
  integer nCtrlID,       // コントロールID
  string strInputTime    // 時刻
  [,float fTimeOut]      // タイムアウト時間
);
```

引数

strCaption（入力用）

コントロールのキャプションを指定してください。

nCtrlID（入力用）

コントロール ID を指定してください。

nHour（入力用）

コントロールに設定する時間を指定してください。

nMinute（入力用）

コントロールに設定する分を指定してください。

nSecond（入力用）

コントロールに設定する秒を指定してください。

strInputTime（入力用）

コントロールに設定する時刻を指定してください。時刻は *hh:mm:ss* 形式で，*hh* が時間，*mm* が分，そして *ss* が秒を示すように指定してください。

fTimeOut（入力用，省略可）

この関数がコントロールを見つけるために使用できる最大時間を秒単位で指定してください。省略した場合は，関数 AIT_SetDefaultWaitTimeout で指定されている値が使われます。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false になります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。

AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
6	ERROR_INVALID_HANDLE
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

注意事項

コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連するラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_SetKeyState

機能

指定したキー状態に設定します。

形式

```
bool AIT_SetKeyState  (
  integer nVirtualKey,      // 仮想キー
  integer nKeyState         // キー状態
);
```

引数

nVirtualKey（入力用）

設定する仮想キーを指定してください。これは，次の値のどれかにする必要があります。

値	意味
NUMLOCK	［Num Lock］キー
SCROLLLOCK	［Scroll Lock］キー
CAPSLOCK	［Caps Lock］キー

nKeyState（入力用）

設定するキー状態を指定してください。これは，次の値のどれかにする必要があります。

値	意味
KEYSTATE_OFF	キーのオフ状態
KEYSTATE_ON	キーのオン状態

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false になります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL

AIT_SetProfileString

機能

INI ファイルのセクションにあるキーの値を作成または変更します。

形式

```
bool AIT_SetProfileString  (
  string strIniFileName,     // INIファイル名
  string strSectionName,     // セクション名
  string strKeyName,         // キー名
  string strValue            // 値
);
```

引数

strIniFileName**（入力用）**

INI ファイル名を指定してください。

strSectionName**（入力用）**

INI ファイルのセクション名を指定してください。

strKeyName**（入力用）**

セクション名に属するキー名を指定してください。

strValue**（入力用）**

設定するキーの値を指定してください。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false になります。

関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
2	ERROR_FILE_NOT_FOUND
3	ERROR_PATH_NOT_FOUND

拡張エラー番号	エラーコード
5	ERROR_ACCESS_DENIED
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
15	ERROR_INVALID_DRIVE
21	ERROR_NOT_READY
23	ERROR_CRC
53	ERROR_BAD_NETPATH
67	ERROR_BAD_NET_NAME
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
123	ERROR_INVALID_NAME
148	ERROR_PATH_BUSY
1005	ERROR_UNRECOGNIZED_VOLUME

AIT_SetScrollPos

機能

スクロールバーを移動します。

形式

```
bool AIT_SetScrollPos (
  integer nCtrlID,          // コントロールID
  integer nPosition         // 位置
  [,float fTimeOut]         // タイムアウト時間
);
bool AIT_SetScrollPos (
  string strCaption,        // コントロールのキャプション
  integer nCtrlType,        // コントロールタイプ
  integer nScrollType,      // スクロールタイプ
  integer nPosition,        // 設定位置
  integer nScrollMovement   // 移動タイプ
  [,float fTimeOut]         // タイムアウト時間
);
bool AIT_SetScrollPos (
  integer nCtrlID,          // コントロールID
  integer nCtrlType,        // コントロールタイプ
  integer nScrollType,      // スクロールタイプ
  integer nPosition,        // 設定位置
  integer nScrollMovement   // 移動タイプ
  [,float fTimeOut]         // タイムアウト時間
);
```

引数

nCtrlID（入力用）

コントロール ID を指定してください。

nPosition（入力用）

設定する位置を指定してください。

strCaption（入力用）

コントロールのキャプションを指定してください。

nCtrlType（入力用）

スクロールバーを持つコントロールタイプを指定してください。これは，次の値のどれかにする必要があります。

値	意味
EDIT_CTRL	コントロールタイプはエディットコントロールです。
LISTBOX_CTRL	コントロールタイプはリストボックスです。
LIST_CTRL	コントロールタイプはリストコントロールです。
TREE_CTRL	コントロールタイプはツリーコントロールです。

nScrollType（入力用）

スクロールバーのタイプを指定してください。これは，次の値のどれかにする必要があります。

値	意味
VSCROLL	縦スクロールバー
HSCROLL	横スクロールバー

nScrollMovement（入力用）

スクロールバーコントロールの移動タイプを指定してください。nScrollType が VSCROLL の場合，これは次の値のどれかにする必要があります。

値	意味
SB_BOTTOM	右下にスクロールします。
SB_LINEDOWN	1 行下にスクロールします。
SB_LINEUP	1 行上にスクロールします。
SB_PAGEDOWN	1 ページ下にスクロールします。
SB_PAGEUP	1 ページ上にスクロールします。
SB_THUMBPOSITION	ユーザはスクロールボックス（サム）をドラッグしてマウスボタンを放しました。
SB_THUMBTRACK	ユーザはスクロールボックスをドラッグしています。
SB_TOP	左上にスクロールします。

nScrollType が HSCROLL の場合，これは次の値のどれかにする必要があります。

値	意味
SB_LEFT	左上にスクロールします。
SB_RIGHT	右下にスクロールします。
SB_LINELEFT	1 単位左にスクロールします。
SB_LINERIGHT	1 単位右にスクロールします。
SB_PAGELEFT	ウィンドウ幅分左にスクロールします。

値	意味
SB_PAGERIGHT	ウィンドウ幅分右にスクロールします。
SB_THUMBPOSITION	ユーザはスクロールボックス（サム）をドラッグしてマウスボタンを放しました。
SB_THUMBTRACK	ユーザはスクロールボックスをドラッグしています。

nTimeOut**（入力用，省略可）**

この関数がコントロールを見つけるために使用できる最大時間を秒単位で指定してください。省略した場合は，関数 AIT_SetDefaultWaitTimeout で設定した値が使用されます。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false になります。関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

注意事項

コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連するラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_SetSpinPos

機能

アクティブなウィンドウの特定のコントロールでの位置を設定します。

形式

```
bool AIT_SetSpinPos (
  string strCaption,      // コントロールのキャプション
  integer nCtrlType,      // コントロールタイプ
  integer nPosition       // 設定位置
  [,float fTimeOut]       // タイムアウト時間
);
bool AIT_SetSpinPos (
  integer nCtrlID,        // コントロールID
  integer nCtrlType,      // コントロールタイプ
  integer nPosition       // 設定位置
  [,float fTimeOut]       // タイムアウト時間
);
```

引数

strCaption（入力用）

コントロールのキャプションを指定してください。

nCtrlID（入力用）

コントロール ID を指定してください。

nCtrlType（入力用）

コントロールタイプを指定してください。これは，次の値のどれかにする必要があります。

値	意味
SPIN_CTRL	コントロールタイプがスピンコントロール
SLIDER_CTRL	コントロールタイプがスライダーコントロール

nPosition（入力用）

設定する位置を指定してください。

fTimeOut（入力用，省略可）

この関数がコントロールを見つけるために使用できる最大時間を秒単位で指定してください。省略した場合は，関数 AIT_SetDefaultWaitTimeout で設定した値が使用されます。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false になります。関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

注意事項

コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連するラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_SetWndPos

機能

指定したウィンドウの位置を変更します。

形式

```
bool AIT_SetWndPos  (
  integer nWndHandle,      // ウィンドウハンドル
  integer nLeft,           // 水平位置
  integer nTop             // 垂直位置
);
```

引数

nWndHandle（入力用）

ウィンドウのハンドルを指定してください。0 を指定した場合，アクティブなウィンドウの位置が設定されます。

nLeft（入力用）

変更後のウィンドウの左上隅の X 座標（水平位置）を指定してください。

nTop（入力用）

変更後のウィンドウの左上隅の Y 座標（垂直位置）を指定してください。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false になります。関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE

AIT_SetWndPosSize

機能

指定したウィンドウの位置とサイズを変更します。

形式

```
bool AIT_SetWndPosSize  (
  integer nHandle,       // ウィンドウハンドル
  integer nLeft,         // 水平位置
  integer nTop,          // 垂直位置
  integer nWidth,        // ウィンドウの幅
  integer nHeight        // ウィンドウの高さ
);
```

引数

nHandle（入力用）

ウィンドウのハンドルを指定してください。0 を指定した場合，アクティブなウィンドウの位置が設定されます。

nLeft（入力用）

変更後のウィンドウの左上隅の X 座標（水平位置）を指定してください。

nTop（入力用）

変更後のウィンドウの左上隅の Y 座標（垂直位置）を指定してください。

nWidth（入力用）

変更後のウィンドウの幅を指定してください。

nHeight（入力用）

変更後のウィンドウの高さを指定してください。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false になります。関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE

AIT_Sleep

機能

AIT ファイルの実行を一定時間停止します。

形式

```
AIT_Sleep  (
  float fSeconds        // 秒数
);
```

引数

fSeconds（入力用）

実行を停止する時間を秒単位で指定してください。

戻り値

なし

AIT_StatusBox

機能

入力したメッセージを含むダイアログボックスを表示します。

形式

```
bool AIT_StatusBox(
  string       strMessage,   // メッセージ文字列
  [[[[,integer nXCord]       // X座標
  [,integer    nYCord]       // Y座標
  [,integer    nWidth]       // メッセージボックスの幅
  [,integer    nHeight]]     // メッセージボックスの高さ
  [,bool       bIsTop]       // 最前面のメッセージボックス
  [,bool       bIsMovable]]  // 移動可能なメッセージボックス
  [,string     strFontName]] // メッセージのフォント名
  [,integer    nFontSize]    // メッセージのフォントサイズ
  [,integer    nFontWeight]  // メッセージのフォント幅
);
```

引数

strMessage（入力用）

ダイアログボックスに表示するメッセージ文字列を指定してください。

nXCord（入力用，省略可）

ダイアログボックスの左上隅の X 座標を指定してください。-1 を指定した場合，X 座標の中央にダイアログボックスが表示されます。

nYCord（入力用，省略可）

ダイアログボックスの左上隅の Y 座標を指定してください。-1 を指定した場合，Y 座標の中央にダイアログボックスが表示されます。

nXCord および nYCord を省略した場合は，ダイアログボックスが中央に配置されます。

nWidth（入力用，省略可）

ダイアログボックスの幅を指定してください。

nHeight（入力用，省略可）

ダイアログボックスの高さを指定してください。

nWidth および nHeight を省略した場合は，ダイアログボックスのサイズが strMessage のサイズと同じになります。

bIsTop（入力用，省略可）

true を指定すると，ダイアログボックスは常に最前面に配置されます。false を指定すると，最初は前面に表示されますが，ほかのウィンドウを移動したり作成したりすると，背面に移動します。任意の動作には false を指定してください。

なお，true を指定しても，ほかのウィンドウの動作によってダイアログボックスが背面に移動する場合があります。この場合，再度 API を実行することでダイアログボックスを最前面に表示できます。

この引数を使用する場合は，nXCord，nYCord，nWidth，および nHeight を指定する必要があります。

bIsMovable（入力用，省略可）

true を指定すると，ダイアログボックスは移動可能になります。false を指定すると，ダイアログボックスは移動できません。

この引数を使用する場合は，nXCord，nYCord，nWidth，および nHeight を指定する必要があります。

strFontName（入力用，省略可）

メッセージの表示フォントを指定する文字列を指定してください。使用可能なフォントは，システムによって異なることがあります。

任意の動作には "" を指定してください。この場合，デフォルトフォントはシステムが決定します。

この引数を使用する場合は，nXCord，nYCord，nWidth，nHeight，bIsTop，および bIsMovable を指定する必要があります。

nFontSize（入力用，省略可）

メッセージのフォントサイズを整数で指定してください。単位はポイントです。

任意の動作には 0 を指定してください。この場合，デフォルトフォントサイズはシステムが決定します。

nFontWeight（入力用，省略可）

メッセージのフォント幅を整数で指定してください。有効な値は 0 ～ 900 です。最低の値を指定すると最も細いフォントに，最高の値を指定すると最も太いフォントになります。

任意の動作には 0 を指定してください。この場合，デフォルトのフォントの太さはシステムが決定します。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false になります。関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
87	ERROR_INVALID_PARAMETER

注意事項

省略可と示されている引数は独立した引数ではありません。その前の省略可能な引数をデフォルト値で指定する必要があります。例えば nXCord と nYCord だけを指定する場合は，あとに続くほかの省略可能な引数を省けます。しかし，nFontSize と nFontWeight だけを指定する場合は，あとに続く省略可能な引数を省くことはできません。その前の省略可能な引数にはデフォルト値を指定する必要があります。

AIT_StatusBoxClose

機能

表示されているステータスボックスを閉じます。

形式

```
AIT_StatusBoxClose ();
```

引数

なし

戻り値

なし

AIT_StrLeft

機能

文字列の左から指定した文字数分だけ返します。

strStrName の先頭（左端）から，nNumChars で指定した文字数分を抽出します。nNumChars が文字列の長さを超える場合は，文字列全体を抽出します。

形式

```
string AIT_StrLeft (
  string strStrName,    // 文字列
  integer nNumChars     // 文字数
);
```

引数

strStrName**（入力用）**

文字列を指定してください。

nNumChars**（入力用）**

抽出する文字数を指定してください。

戻り値

抽出した文字列を返します。

注意事項

マルチバイト文字セット（MBCS）の場合は，8 ビットごとに 1 文字としてカウントします。つまり，1 マルチバイト文字の先頭および末尾のバイトを 2 文字としてカウントします。

AIT_StrLength

機能

文字列の長さを返します。

形式

```
integer AIT_StrLength (
  string strStrName    // 文字列
);
```

引数

strStrName**（入力用）**

文字列を指定してください。

戻り値

文字列の長さを返します。

AIT_StrLower

機能

文字列を小文字に変換します。

形式

```
string AIT_StrLower (
  string strStrName    // 文字列
);
```

引数

strStrName**（入力用）**

文字列を指定してください。

戻り値

小文字に変換した文字列を返します。

AIT_StrLTrim

機能

文字列の先頭（左端）から，空白または指定した文字をすべて削除した文字列を返します。

形式

```
string AIT_StrLTrim (
  string strStrName        // 文字列
  [,string strCharValue]   // 文字の値
);
```

引数

strStrName**（入力用）**

文字列を指定してください。

strCharValue**（入力用，省略可）**

削除する文字を指定してください。省略した場合，文字列の先頭から空白（改行，スペース，タブ文字など）を削除します。

戻り値

文字列の先頭からすべての空白または指定した文字を削除した文字列を返します。

AIT_StrRight

機能

文字列の右から，指定した文字数分だけ返します。

strStrName の末尾（右端）から，nNumChars で指定した文字数分を抽出します。nNumChars が文字列の長さを超える場合は，文字列全体を返します。

形式

```
string AIT_StrRight (
  string strStrName,    // 文字列
  integer nNumChars     // 文字数
);
```

引数

strStrName**（入力用）**

文字列を指定してください。

nNumChars**（入力用）**

抽出する文字数を指定してください。

戻り値

抽出した文字列を返します。

注意事項

マルチバイト文字セット（MBCS）の場合は，8 ビットごとに 1 文字としてカウントします。つまり，1 マルチバイト文字の先頭および末尾のバイトを 2 文字としてカウントします。

AIT_StrRTrim

機能

文字列の末尾（右端）から空白または指定した文字をすべて削除した文字列を返します。

形式

```
string AIT_StrRTrim (
  string strStrName        // 文字列
  [,string strCharValue]   // 文字の値
);
```

引数

strStrName**（入力用）**

文字列を指定してください。

strCharValue**（入力用，省略可）**

削除する文字を指定してください。省略した場合は，文字列の末尾から空白（改行，スペース，タブ文字など）を削除します。

戻り値

文字列の末尾からすべての空白または指定した文字を削除した文字列を返します。

AIT_StrTrim

機能

文字列の先頭（左端）および，末尾（右端）から空白または指定した文字をすべて削除した文字列を返します。

形式

```
string AIT_StrTrim (
  string strStrName         // 文字列
  [,string strCharValue]    // 文字の値
);
```

引数

strStrName**（入力用）**

文字列を指定してください。

strCharValue**（入力用，省略可）**

削除する文字を指定してください。省略した場合は，先頭，および末尾から空白（改行，スペース，タブ文字など）を削除します。

戻り値

文字列の先頭，および末尾から空白または指定した文字を削除した文字列を返します。

AIT_StrUpper

機能

文字列を大文字に変換します。

形式

```
string AIT_StrUpper (
  string strStrName    // 文字列
);
```

引数

strStrName**（入力用）**

文字列を指定してください。

戻り値

大文字に変換した文字列を返します。

AIT_TaskbarClk

機能

タスクバーの空いている場所をクリックします。

形式

```
bool AIT_TaskbarClk  (
  [integer nMouseButton]     // マウスボタン
);
```

引数

nMouseButton**（入力用，省略可）**

マウスでクリックするボタンを指定してください。これは，次の値のどれかにする必要があります。

値	意味
LBUTTON	左マウスボタン
MBUTTON	中央マウスボタン
RBUTTON	右マウスボタン

省略した場合は，RBUTTON がデフォルト値として使用されます。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false になります。関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
1400	ERROR_INVALID_WINDOW_HANDLE

AIT_TaskbarHasFocus

機能

タスクバーが入力フォーカスを持っているかどうかを確認します。

形式

```
integer AIT_TaskbarHasFocus ();
```

引数

なし

戻り値

タスクバーがフォーカスを持っている場合は戻り値が 1，タスクバーがフォーカスを持っていない場合は戻り値が 0，失敗した場合は戻り値が -1 になります。-1 が返された場合は，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
1400	ERROR_INVALID_WINDOW_HANDLE

AIT_TaskbarItemClk

機能

タスクバーの特定のタブ項目上をクリックします。

形式

```
bool AIT_TaskbarItemClk (
  integer nIndex            // インデックス
  [,integer nMouseButton]   // マウスボタン
);
```

引数

nIndex**（入力用）**

タスクバー項目のインデックスを指定してください。インデックスは 1 を基準とするため，最初の項目のインデックスは 1 になります。

nMouseButton**（入力用，省略可）**

マウスでクリックするボタンを指定してください。これは，次の値のどれかにする必要があります。

値	意味
LBUTTON	左マウスボタン
MBUTTON	中央マウスボタン
RBUTTON	右マウスボタン

省略した場合は，LBUTTON がデフォルト値として使用されます。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false になります。関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
1400	ERROR_INVALID_WINDOW_HANDLE
1413	ERROR_INVALID_INDEX

注意事項

この関数は，Windows 2000，Windows NT 4.0，Windows Me，および Windows 98 で使用できます。Windows 8，Windows Server 2012，Windows 7，Windows Server 2008，Windows Vista，Windows Server 2003，および Windows XP では使用できません。

AIT_TaskbarItemExists

機能

タスクバーに特定の項目が存在するかどうかを確認します。

形式

```
integer AIT_TaskbarItemExists  (
  integer nIndex      // インデックス
);
```

引数

nIndex（入力用）

タスクバー項目のインデックスを指定してください。インデックスは 1 を基準とするため，最初の項目の nIndex は 1 になります。

戻り値

タスクバー項目が存在する場合の戻り値は 1，タスクバー項目が存在しない場合の戻り値は 0，失敗した場合の戻り値は -1 になります。-1 が返された場合は，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
1400	ERROR_INVALID_WINDOW_HANDLE
1413	ERROR_INVALID_INDEX

注意事項

この関数は，Windows 2000，Windows NT 4.0，Windows Me，および Windows 98 で使用できます。Windows 8，Windows Server 2012，Windows 7，Windows Server 2008，Windows Vista，Windows Server 2003，および Windows XP では使用できません。

AIT_TaskbarItemIndex

機能

タスクバー上の選択されている項目のインデックスを返します。

形式

```
bool AIT_TaskbarItemIndex  (
  integer nIndex      // インデックス
);
```

引数

nIndex（出力用）

タスクバー項目のインデックスを受け取る変数を指定してください。インデックスは 1 を基準にしているため，最初の項目のインデックスは 1 になります。

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false になります。関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
1400	ERROR_INVALID_WINDOW_HANDLE

注意事項

この関数は，Windows 2000，Windows NT 4.0，Windows Me，および Windows 98 で使用できます。Windows 8，Windows Server 2012，Windows 7，Windows Server 2008，Windows Vista，Windows Server 2003，および Windows XP では使用できません。

AIT_TaskbarItemSelected

機能

タスクバーの特定の項目が選択されているかどうかを確認します。

形式

```
integer AIT_TaskbarItemSelected  (
  integer nIndex      // インデックス
);
```

引数

nIndex（入力用）

タスクバー項目のインデックスを指定してください。インデックスは 1 を基準にしているため，最初の項目のインデックスは 1 になります。

戻り値

タスクバー項目が選択されている場合の戻り値は 1，タスクバー項目が選択されていない場合の戻り値は 0，失敗した場合の戻り値は -1 になります。-1 が返された場合は，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
6	ERROR_INVALID_HANDLE
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
1413	ERROR_INVALID_INDEX

注意事項

この関数は，Windows 2000，Windows NT 4.0，Windows Me，および Windows 98 で使用できます。Windows 8，Windows Server 2012，Windows 7，Windows Server 2008，Windows Vista，Windows

Server 2003，および Windows XP では使用できません。

AIT_TaskbarSetFocus

機能

タスクバーに入力フォーカスを設定します。

形式

```
bool AIT_TaskbarSetFocus ();
```

引数

なし

戻り値

関数が正常に処理された場合の戻り値は true，そのほかの場合は false になります。関数が false を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
1400	ERROR_INVALID_WINDOW_HANDLE

AIT_VerifyCharPos

機能

アクティブなウィンドウの特定のコントロールで文字位置を確認します。

形式

```
integer AIT_VerifyCharPos  (
  string strCaption,      // コントロールのキャプション
  integer nCtrlType,      // コントロールタイプ
  integer nVerifyPos      // 文字位置
);
integer AIT_VerifyCharPos  (
  integer nCtrlID,        // コントロールID
  integer nCtrlType,      // コントロールタイプ
  integer nVerifyPos      // 文字位置
);
```

引数

strCaption**（入力用）**

コントロールのキャプションを指定してください。

nCtrlID**（入力用）**

コントロール ID を指定してください。

nCtrlType（入力用）

コントロールタイプを指定してください。コントロールタイプは EDIT_CTRL だけが有効です。

nVerifyPos（入力用）

確認する文字の位置の値を指定してください。位置は 0 を基準にしているため，エディットボックスの最初の文字は，位置が 0 になります。

戻り値

検索した位置が指定した位置と同じ場合の戻り値は 1，異なる場合の戻り値は 0，失敗した場合の戻り値は -1 になります。

-1 が返された場合は，AIT_GetLastError を使用して拡張エラーコードを取得できます。
AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

注意事項

コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連するラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_VerifyCount

機能

アクティブなウィンドウの特定のコントロールで項目数を確認します。

形式

```
integer AIT_VerifyCount  (
  string strCaption,     // コントロールのキャプション
  integer nCtrlType,     // コントロールタイプ
  integer nItemCount     // 項目数
);
integer AIT_VerifyCount  (
  integer nCtrlID,       // コントロールID
  integer nCtrlType,     // コントロールタイプ
  integer nItemCount     // 項目数
);
```

引数

strCaption（入力用）

コントロールのキャプションを指定してください。

nCtrlID（入力用）

コントロール ID を指定してください。

nCtrlType（入力用）

コントロールタイプを指定してください。これは，次の値のどれかにする必要があります。

値	意味
COMBO_CTRL	コントロールタイプがコンボボックスです。
LISTBOX_CTRL	コントロールタイプがリストボックスです。
TREE_CTRL	コントロールタイプがツリーコントロールです。
LIST_CTRL	コントロールタイプがリストコントロールです。

nItemCount（入力用）

確認する項目数の値を指定してください。

戻り値

指定した項目数と一致する場合の戻り値は 1，一致しない場合の戻り値は 0，失敗した場合の戻り値は -1 になります。

-1 が返された場合は，AIT_GetLastError を使用して拡張エラーコードを取得できます。
AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
6	ERROR_INVALID_HANDLE
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

注意事項

コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連するラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_VerifyDateTime

機能

アクティブなウィンドウの特定のコントロールで日付または時間を確認します。

形式

```
integer AIT_VerifyDateTime  (
  string strCaption,     // コントロールのキャプション
```

```
  string strDateTime     // 日付または時間
);
integer AIT_VerifyDateTime  (
  integer nCtrlID,       // コントロールID
  string strDateTime     // 日付または時間
);
```

引数

strCaption（入力用）

コントロールのキャプションを指定してください。

nCtrlID（入力用）

コントロール ID を指定してください。

strDateTime（入力用）

コントロールの日付または時間を指定してください。

戻り値

指定した日付または時間と一致した場合の戻り値は 1，一致しない場合の戻り値は 0，失敗した場合の戻り値は -1 となります。

関数が -1 を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。
AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

注意事項

コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連するラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_VerifyDefaultButton

機能

アクティブなウィンドウで特定のコマンドボタンがデフォルトボタンかどうかを確認します。

形式

```
integer AIT_VerifyDefaultButton  (
  string strCaption    // コントロールのキャプション
);
integer AIT_VerifyDefaultButton  (
  integer nCtrlID      // コントロールID
);
```

引数

strCaption（入力用）

コントロールのキャプションを指定してください。

nCtrlID（入力用）

コントロール ID を指定してください。

戻り値

ボタンがデフォルトボタンの場合の戻り値は 1，デフォルトボタンではない場合の戻り値は 0，失敗した場合の戻り値は -1 となります。

関数が -1 を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

注意事項

コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連するラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_VerifyEnabled

機能

アクティブなウィンドウの特定のコントロールが使用可能かを確認します。

形式

```
integer AIT_VerifyEnabled  (
  string strCaption,     // コントロールのキャプション
  integer nCtrlType      // コントロールタイプ
  [,integer nCtrlPos]    // タブオーダー
);
integer AIT_VerifyEnabled  (
  integer nCtrlID,       // コントロールID
  integer nCtrlType      // コントロールタイプ
  [,integer nCtrlPos]    // タブオーダー
);
```

引数

strCaption（入力用）

コントロールのキャプションを指定してください。

nCtrlID（入力用）

コントロール ID を指定してください。

nCtrlType（入力用）

コントロールタイプを指定してください。次の値のどれかに該当する必要があります。

値	意味
BUTTON_CTRL	コントロールタイプはコマンドボタンです。
CHECKBOX_CTRL	コントロールタイプはチェックボックスです。
OPTIONBUTTON_CTRL	コントロールタイプはオプションボタンです。
EDIT_CTRL	コントロールタイプはエディットボックスです。
STATIC_CTRL	コントロールタイプはスタティックテキストです。
COMBO_CTRL	コントロールタイプはコンボボックスです。
LISTBOX_CTRL	コントロールタイプはリストボックスです。
SPIN_CTRL	コントロールタイプはスピンコントロールです。
TREE_CTRL	コントロールタイプはツリーコントロールです。
LIST_CTRL	コントロールタイプはリストコントロールです。
DTPICKER_CTRL	コントロールタイプは日時ピッカーです。

nCtrlPos（入力用，省略可）

コントロールのタブオーダーを指定してください。

戻り値

コントロールが使用可能な場合の戻り値は 1，使用可能ではない場合の戻り値は 0，失敗した場合の戻り値は -1 となります。

関数が -1 を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

注意事項

コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連するラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_VerifyExistence

機能

アクティブなウィンドウで特定のコントロールが存在するかを確認します。

形式

```
integer AIT_VerifyExistence  (
  string strCaption,    // コントロールのキャプション
  integer nCtrlType     // コントロールタイプ
);
integer AIT_VerifyExistence  (
  integer nCtrlID,      // コントロールID
  integer nCtrlType     // コントロールタイプ
);
```

引数

strCaption（入力用）

コントロールのキャプションを指定してください。

nCtrlID（入力用）

コントロール ID を指定してください。

nCtrlType（入力用）

コントロールタイプを指定してください。次の値のどれかに該当する必要があります。

値	意味
BUTTON_CTRL	コントロールタイプはコマンドボタンです。
CHECKBOX_CTRL	コントロールタイプはチェックボックスです。
OPTIONBUTTON_CTRL	コントロールタイプはオプションボタンです。
EDIT_CTRL	コントロールタイプはエディットボックスです。
STATIC_CTRL	コントロールタイプはスタティックテキストです。
COMBO_CTRL	コントロールタイプはコンボボックスです。
LISTBOX_CTRL	コントロールタイプはリストボックスです。
SPIN_CTRL	コントロールタイプはスピンコントロールです。
TREE_CTRL	コントロールタイプはツリーコントロールです。
LIST_CTRL	コントロールタイプはリストコントロールです。
DTPICKER_CTRL	コントロールタイプは日時ピッカーです。
IPADDRESS_CTRL	コントロールタイプは IP アドレスコントロールです。
SLIDER_CTRL	コントロールタイプはスライダーコントロールです。
SCROLLBAR_CTRL	コントロールタイプはスクロールバーコントロールです。

戻り値

コントロールが存在する場合の戻り値は 1，存在しない場合の戻り値は 0，失敗した場合の戻り値は -1 となります。

関数が -1 を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。
AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1460	ERROR_TIMEOUT

注意事項

コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連するラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_VerifyFirstVisible

機能

リストボックスの最初の，可視項目のインデックスを確認します。

形式

```
integer AIT_VerifyFirstVisible  (
  string strCaption,     // コントロールのキャプション
  integer nCtrlType,     // コントロールタイプ
  integer nIndex         // インデックス
);
integer AIT_VerifyFirstVisible  (
  integer nCtrlID,       // コントロールID
  integer nCtrlType,     // コントロールタイプ
  integer nIndex         // インデックス
);
```

引数

strCaption**（入力用）**

コントロールのキャプションを指定してください。

nCtrlID**（入力用）**

コントロール ID を指定してください。

nCtrlType**（入力用）**

コントロールタイプを指定してください。コントロールタイプは LISTBOX_CTRL だけ有効です。

nIndex**（入力用）**

コントロールのインデックスを指定してください。インデックスの基準値は 0 で，コントロールの最初の項目でのインデックスは 0 になります。

戻り値

指定したインデックスが最初の可視項目のインデックスと一致する場合の戻り値は 1，一致しない場合の戻り値は 0，失敗した場合の戻り値は -1 となります。

関数が -1 を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。

AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

注意事項

コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連するラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_VerifyFocus

機能

アクティブなウィンドウの特定のコントロールがフォーカスを持っているかどうかを確認します。

形式

```
integer AIT_VerifyFocus  (
  string strCaption,     // コントロールのキャプション
  integer nCtrlType      // コントロールタイプ
);
integer AIT_VerifyFocus  (
  integer nCtrlID,       // コントロールID
  integer nCtrlType      // コントロールタイプ
);
```

引数

strCaption**（入力用）**

コントロールのキャプションを指定してください。

nCtrlID**（入力用）**

コントロール ID を指定してください。

nCtrlType**（入力用）**

コントロールタイプを指定してください。次の値のどれかを指定します。

値	意味
BUTTON_CTRL	コントロールタイプはコマンドボタンです。
CHECKBOX_CTRL	コントロールタイプはチェックボックスです。
OPTIONBUTTON_CTRL	コントロールタイプはオプションボタンです。
EDIT_CTRL	コントロールタイプはエディットボックスです。
STATIC_CTRL	コントロールタイプはスタティックテキストです。

値	意味
COMBO_CTRL	コントロールタイプはコンボボックスです。
LISTBOX_CTRL	コントロールタイプはリストボックスです。
SPIN_CTRL	コントロールタイプはスピンコントロールです。
TREE_CTRL	コントロールタイプはツリーコントロールです。
LIST_CTRL	コントロールタイプはリストコントロールです。
DTPICKER_CTRL	コントロールタイプは日時ピッカーです。

戻り値

コントロールがフォーカスを持っている場合の戻り値は 1，フォーカスを持っていない場合の戻り値は 0，失敗した場合の戻り値は -1 となります。

関数が -1 を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

注意事項

コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連するラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_VerifyIndex

機能

アクティブなウィンドウの特定のコントロールでテキストとインデックスが一致しているかどうかを確認します。

形式

```
integer AIT_VerifyIndex  (
  string strCaption,     // コントロールのキャプション
  integer nCtrlType,     // コントロールタイプ
  string strCtrlText,    // コントロールテキスト
  integer nIndex         // インデックス
);
integer AIT_VerifyIndex  (
  integer nCtrlID,       // コントロールID
  integer nCtrlType,     // コントロールタイプ
  string strCtrlText,    // コントロールテキスト
  integer nIndex         // インデックス
);
```

引数

strCaption（入力用）

コントロールのキャプションを指定してください。

nCtrlID（入力用）

コントロール ID を指定してください。

nCtrlType（入力用）

コントロールタイプを指定してください。次の値のどれかに該当する必要があります。

値	意味
COMBO_CTRL	コントロールタイプはコンボボックスです。
LISTBOX_CTRL	コントロールタイプはリストボックスです。
TREE_CTRL	コントロールタイプはツリーコントロールです。
LIST_CTRL	コントロールタイプはリストコントロールです。

strCtrlText（入力用）

コントロールのテキストを指定してください。

nIndex（入力用）

コントロールのインデックスを指定してください。インデックスの基準値は 0 です。

戻り値

指定したコントロールのインデックスとテキストが一致する場合の戻り値は 1，一致しない場合の戻り値は 0，失敗した場合の戻り値は -1 となります。

関数が -1 を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。
AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

注意事項

コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連するラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_VerifyKeyState

機能

キーの状態がキーボード上のキーの状態と同じ状態であるかを確認します。

形式

```
integer AIT_VerifyKeyState  (
  integer nVirtualKey,      // 仮想キー
  integer nKeyState         // キーの状態
);
```

引数

nVirtualKey（入力用）

確認するキー状態の仮想キーを指定してください。

次の値のどれかに該当する必要があります。

値	意味
NUMLOCK	［Num Lock］キー
SCROLLLOCK	［Scroll Lock］キー
CAPSLOCK	［Caps Lock］キー

nKeyState（入力用）

確認するキーのオンまたはオフの状態を指定してください。次の値のどれかに該当する必要があります。

値	意味
KEYSTATE_OFF	キーはオフ状態
KEYSTATE_ON	キーはオン状態

戻り値

キーの状態が指定した状態と一致している場合の戻り値は 1，一致していない場合の戻り値は 0，失敗した場合の戻り値は -1 となります。

関数が -1 を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。
AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

AIT_VerifyLine

機能

アクティブなウィンドウでインデックスと複数行のエディットボックス上のカレント行が一致しているかどうかを確認します。

形式

```
integer AIT_VerifyLine  (
  string strCaption,     // コントロールのキャプション
  integer nCtrlType,     // コントロールタイプ
  integer nIndex         // インデックス
);
integer AIT_VerifyLine  (
  integer nCtrlID,       // コントロールID
  integer nCtrlType,     // コントロールタイプ
  integer nIndex         // インデックス
);
```

引数

strCaption（入力用）

コントロールのキャプションを指定してください。

nCtrlID（入力用）

コントロール ID を指定してください。

nCtrlType（入力用）

コントロールタイプを指定してください。コントロールタイプは EDIT_CTRL だけ有効です。

nIndex（入力用）

確認するカレント行のインデックスを指定してください。インデックスの基準値は 0 で，エディットボックスの最初の行でのインデックスは 0 になります。

戻り値

エディットボックスでのカレント行が指定したインデックスと一致する場合の戻り値は 1，一致しない場合の戻り値は 0，失敗した場合の戻り値は -1 となります。

関数が -1 を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。
AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
6	ERROR_INVALID_HANDLE
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

注意事項

コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連するラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_VerifyLocation

機能

アクティブなウィンドウの特定のコントロールが指定した座標と一致するかどうかを確認します。

形式

```
integer AIT_VerifyLocation  (
  string  strCaption,   // コントロールのキャプション
  integer nCtrlType,    // コントロールタイプ
  integer nLeft,        // コントロールの左座標
  integer nTop,         // コントロールの上座標
  integer nRight,       // コントロールの右座標
  integer nBottom       // コントロールの下座標
);
integer AIT_VerifyLocation  (
  integer nCtrlID,      // コントロールID
  integer nCtrlType,    // コントロールタイプ
  integer nLeft,        // コントロールの左座標
  integer nTop,         // コントロールの上座標
  integer nRight,       // コントロールの右座標
  integer nBottom       // コントロールの下座標
);
```

引数

strCaption**（入力用）**

コントロールのキャプションを指定してください。

nCtrlID**（入力用）**

コントロール ID を指定してください。

nCtrlType**（入力用）**

コントロールタイプを指定してください。次の値のどれかに該当する必要があります。

値	意味
BUTTON_CTRL	コントロールタイプはコマンドボタンです。
CHECKBOX_CTRL	コントロールタイプはチェックボックスです。
OPTIONBUTTON_CTRL	コントロールタイプはオプションボタンです。
EDIT_CTRL	コントロールタイプはエディットボックスです。
STATIC_CTRL	コントロールタイプはスタティックテキストです。
COMBO_CTRL	コントロールタイプはコンボボックスです。
LISTBOX_CTRL	コントロールタイプはリストボックスです。
SPIN_CTRL	コントロールタイプはスピンコントロールです。
TREE_CTRL	コントロールタイプはツリーコントロールです。
LIST_CTRL	コントロールタイプはリストコントロールです。

値	意味
DTPICKER_CTRL	コントロールタイプは日時ピッカーです。

nLeft（入力用）

コントロールの左上隅の X 座標を指定してください。

nTop（入力用）

コントロールの左上隅の Y 座標を指定してください。

nRight（入力用）

コントロールの右下隅の X 座標を指定してください。

nBottom（入力用）

コントロールの右下隅の Y 座標を指定してください。

戻り値

コントロールの座標が指定した座標と一致した場合の戻り値は 1，一致しない場合の戻り値は 0，失敗した場合の戻り値は -1 となります。

関数が -1 を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

注意事項

コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連するラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_VerifyMenuChecked

機能

メニュー項目がチェックされているかどうかを確認します。

形式

```
integer AIT_VerifyMenuChecked  (
  integer nMenu,       // メニューハンドル
  integer nIndex       // メニュー項目のインデックス
);
```

引数

nMenu（入力用）

AIT_GetMenu または AIT_GetSubMenu の API が返したメニューハンドルを指定してください。

nIndex（入力用）

メニュー項目のインデックスを指定してください。メニューのインデックス基準値は 0 で，最初のメニュー項目のインデックスは 0 になります。

インデックスにはメニューセパレータのインデックスも含まれます。メニューセパレータのインデックスが入力値として与えられた場合，この関数の処理は失敗し拡張エラーコードで ERROR_INVALID_INDEX を返します。

戻り値

メニュー項目がチェックされている場合の戻り値は 1，チェックされていない場合の戻り値は 0，失敗した場合の戻り値は -1 となります。

関数が -1 を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1401	ERROR_INVALID_MENU_HANDLE
1413	ERROR_INVALID_INDEX
1460	ERROR_TIMEOUT

AIT_VerifyMenuEnabled

機能

メニュー項目が使用可能かどうかを確認します。

形式

```
integer AIT_VerifyMenuEnabled  (
  integer nMenu,      // メニューハンドル
  integer nIndex      // メニュー項目のインデックス
);
```

引数

nMenu（入力用）

AIT_GetMenu または AIT_GetSubMenu の API が返したメニューハンドルを指定してください。

nIndex（入力用）

メニュー項目のインデックスを指定してください。メニュー項目のインデックスは 0 から始まります。したがって，最初のメニュー項目のインデックスは 0 になります。

インデックスには，メニューセパレータのインデックスも含まれます。メニューセパレータのインデックスが入力値として与えられた場合，この関数の処理は失敗し拡張エラーコード ERROR_INVALID_INDEX を返します。

戻り値

メニュー項目が使用可能な場合の戻り値は 1，使用可能ではない場合の戻り値は 0，失敗した場合の戻り値は -1 となります。

関数が -1 を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1401	ERROR_INVALID_MENU_HANDLE
1413	ERROR_INVALID_INDEX
1460	ERROR_TIMEOUT

AIT_VerifyNoOfCtrls

機能

アクティブなウィンドウ内に存在するコントロールの個数が，指定した個数と一致するかどうかを確認します。

形式

```
integer AIT_VerifyNoOfCtrls  (
  integer nNumControls      // コントロール個数
);
```

引数

nNumControls**（入力用）**

確認するコントロールの個数を指定してください。

戻り値

コントロールの個数が一致した場合の戻り値は 1，一致しなかった場合の戻り値は 0，失敗した場合の戻り値は -1 となります。

関数が -1 を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER

拡張エラー番号	エラーコード
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

AIT_VerifyPos

機能

アクティブなウィンドウの特定のコントロールが，指定したタブオーダーと一致するかどうかを確認します。

形式

```
integer AIT_VerifyPos  (
  string strCaption,     // コントロールのキャプション
  integer nCtrlType,     // コントロールタイプ
  integer nCtrlPos       // タブオーダー
);
integer AIT_VerifyPos  (
  integer nCtrlID,      // コントロールID
  integer nCtrlType,    // コントロールタイプ
  integer nCtrlPos      // タブオーダー
);
```

引数

strCaption（入力用）

コントロールのキャプションを指定してください。

nCtrlID（入力用）

コントロール ID を指定してください。

nCtrlType（入力用）

コントロールタイプを指定してください。次の値のどれかに該当する必要があります。

値	意味
BUTTON_CTRL	コントロールタイプはコマンドボタンです。
CHECKBOX_CTRL	コントロールタイプはチェックボックスです。
OPTIONBUTTON_CTRL	コントロールタイプはオプションボタンです。
EDIT_CTRL	コントロールタイプはエディットボックスです。
STATIC_CTRL	コントロールタイプはスタティックテキストです。
COMBO_CTRL	コントロールタイプはコンボボックスです。
LISTBOX_CTRL	コントロールタイプはリストボックスです。
SPIN_CTRL	コントロールタイプはスピンコントロールです。
TREE_CTRL	コントロールタイプはツリーコントロールです。
LIST_CTRL	コントロールタイプはリストコントロールです。
DTPICKER_CTRL	コントロールタイプは日時ピッカーです。

nCtrlPos（入力用）

コントロールのタブオーダーを指定してください。

戻り値

コントロールのタブオーダーが指定したタブオーダーと一致する場合の戻り値は 1，一致しない場合の戻り値は 0，失敗した場合の戻り値は -1 となります。

関数が -1 を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。
AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

注意事項

コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連するラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_VerifySelected

機能

アクティブなウィンドウの特定のコントロールで，指定したテキストがコントロール内で選択されているかどうかを確認します。

形式

```
integer AIT_VerifySelected  (
  string strCaption,       // コントロールのキャプション
  integer nCtrlType,       // コントロールタイプ
  string strSelectedText   // 選択テキスト
);
integer AIT_VerifySelected  (
  integer nCtrlID,         // コントロールID
  integer nCtrlType,       // コントロールタイプ
  string strSelectedText   // 選択テキスト
);
```

引数

strCaption（入力用）

コントロールのキャプションを指定してください。

nCtrlID（入力用）

コントロール ID を指定してください。

nCtrlType（入力用）

コントロールタイプを指定してください。コントロールタイプは EDIT_CTRL だけ有効です。

strSelectedText（入力用）

確認するテキストを指定してください。

戻り値

コントロール内で選択されたテキストが指定したテキストと一致した場合の戻り値は 1，一致しない場合の戻り値は 0，失敗した場合の戻り値は -1 となります。

関数が -1 を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
6	ERROR_INVALID_HANDLE
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

注意事項

コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連するラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_VerifyState

機能

アクティブなウィンドウで特定のコントロールがチェックされているかどうかを確認します。

形式

```
integer AIT_VerifyState  (
  string strCaption,    // コントロールのキャプション
  integer nCtrlType     // コントロールタイプ
);
integer AIT_VerifyState  (
  integer nCtrlID,      // コントロールID
  integer nCtrlType     // コントロールタイプ
);
```

引数

strCaption（入力用）

コントロールのキャプションを指定してください。

nCtrlID（入力用）

コントロール ID を指定してください。

nCtrlType（入力用）

コントロールタイプを指定してください。次の値のどれかに該当する必要があります。

値	意味
CHECKBOX_CTRL	コントロールタイプはチェックボックスです。
OPTIONBUTTON_CTRL	コントロールタイプはオプションボタンです。

戻り値

コントロールがチェックされた状態の戻り値は 1，チェックされていない状態の戻り値は 0，不確定状態の戻り値は 2 になり，失敗した場合の戻り値は -1 になります。不確定状態は CHECKBOX_CTRL だけに適用されます。

関数が -1 を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

注意事項

コントロールは，完全なキャプションまたは関連するラベル名を使用するか，キャプションまたは関連するラベルの最初の部分を指定することで認識できます。キャプションまたは関連するラベルの最初の部分を指定する場合は，文字列の最初に「~」（波記号）を付けます。

AIT_VerifyText

機能

アクティブなウィンドウで，特定のコントロールが指定したテキストと一致するかどうかを確認します。

形式

```
integer AIT_VerifyText  (
  string strCaption,    // コントロールのキャプション
  integer nCtrlType,    // コントロールタイプ
  string strText        // テキスト
);
integer AIT_VerifyText  (
  integer nCtrlID,      // コントロールID
  integer nCtrlType,    // コントロールタイプ
  string strText        // テキスト
);
```

引数

strCaption（入力用）

コントロールのキャプションを指定してください。

nCtrlID（入力用）

コントロール ID を指定してください。

nCtrlType（入力用）

コントロールタイプを指定してください。コントロールタイプは EDIT_CTRL だけ有効です。

strText（入力用）

確認するテキストを指定してください。

戻り値

コントロールのテキストが指定したテキストと一致する場合の戻り値は 1，一致しない場合の戻り値は 0 となり，失敗した場合の戻り値は -1 となります。

関数が -1 を返した場合には，AIT_GetLastError を使用して拡張エラーコードを取得できます。
AIT_GetLastError が返す可能性のあるエラーコードを次に示します。

拡張エラー番号	エラーコード
8	ERROR_NOT_ENOUGH_MEMORY
14	ERROR_OUTOFMEMORY
87	ERROR_INVALID_PARAMETER
112	ERROR_DISK_FULL
1400	ERROR_INVALID_WINDOW_HANDLE
1460	ERROR_TIMEOUT

4.3 API の使用例

AIT 言語が提供する API の使用例を次に示します。

4.3.1 復帰と改行を削除する

ファイルから読み出した文章には，復帰と改行（¥r¥n）が含まれている場合があります。ファイルから読み出した文章「復帰と改行を削除しました。¥r¥n」から復帰と改行を削除して，RecDFile.log に実行結果を出力する例を次に示します。

(1) 記述例

```
strFileName = "C:'¥Sample.txt";
strCharValue = "¥r¥n";                    // 復帰と改行の文字列
if (AIT_FileOpen(strFileName, GENERIC_READ, OPEN_EXISTING, nFileHandle))
  // ファイルからデータを読み込みます。
  if (!AIT_FileGetLine(nFileHandle, strReadData))
    AIT_LogMessage("AIT_FileGetLine failed");
  else
    // 読み込んだデータから，復帰と改行を削除します。
    strStrName = AIT_StrRTrim(strReadData, strCharValue);
    AIT_LogMessage("strStrName = " + strStrName);
  endif;
  AIT_FileClose(nFileHandle);
else
  AIT_LogMessage("AIT_FileOpen failed");
endif;
```

(2) 実行結果

RecDFile.log への出力結果を次に示します。

```
strStrName = 復帰と改行を削除しました。
```

4.3.2 文字列を抽出する

文字列「0123-4567-89AB-CDEF」から英数字だけを抽出して，RecDFile.log へ実行結果を出力する例を次に示します。

(1) 記述例

```
strStrName = "0123-4567-89AB-CDEF";    // 文字列
strSearchStr = "-";                    // 検索文字列
nStartPos = 0;
while(TRUE)
  // 抽出する文字列の長さを取得します。
  nLength = AIT_FindSubStr(strStrName, strSearchStr, nStartPos);
  if (nLength == -1)
    // 最後に抽出する文字列を設定します。
    strSubString = strStrName;
    AIT_LogMessage("strSubString = " + strSubString);
    // 文字列の抽出を終了します。
    break;
  else
    // 文字列を抽出します。
    if (!AIT_GetSubStr(strSubString, strStrName, nStartPos, nLength))
      AIT_LogMessage("AIT_GetSubStr failed");
      break;
    else
      AIT_LogMessage("strSubString = " + strSubString);
    endif;
    // 抽出する文字列から抽出した文字列を削除します。
    strStrName = AIT_StrLTrim(strStrName, strSubString);
    strStrName = AIT_StrLTrim(strStrName, strSearchStr);
```

```
    strSubString = "";
  endif;
loop;
// strStrNameは処理開始前の値と異なります。
AIT_LogMessage("strStrName = " + strStrName);
```

(2) 実行結果

RecDFile.log への出力結果を次に示します。

```
strSubString = 0123
strSubString = 4567
strSubString = 89AB
strSubString = CDEF
strStrName = CDEF
```

4.3.3 リモートインストール時にレジストリの HKEY_CURRENT_USER を操作する

AIT ファイルでレジストリの HKEY_CURRENT_USER を操作する API を使用する場合，リモートインストール先のクライアントに Administrator 権限を持たないユーザがログオンしているときは，操作するレジストリが HKEY_USERS¥.DEFAULT へ変更されます。このとき，レジストリの HKEY_CURRENT_USER を操作するには，［スタート］メニューから［ファイル名を指定して実行］を実行して，レジストリファイルをインポートする必要があります。

AIT ファイルを使用して，リモートインストール時にレジストリの HKEY_CURRENT_USER を操作する手順を説明します。

1. **レジストリファイルを作成する。**
   追加，変更などを実行したいレジストリのレジストリファイルを作成します。

! 注意事項

レジストリを操作する場合は，レジストリをバックアップしておくことと，システムの復元方法を理解しておくことをお勧めします。

---

2. AIT **ファイルを作成する。**
   ［スタート］メニューから［ファイル名を指定して実行］を実行し，regedit.exe を利用してレジストリファイルをインポートする例を次に示します。この例では，インポートするレジストリファイルを「Sample.reg」としています。

```
while(iLoopCount < iLoopMax)
  if((AITEVENTFLAG1==0) && (AITIGNORE == 0))
    // ［スタート］メニューを表示します。
    AIT_PlayKey("{LWIN}");
    AIT_Sleep(SLEEP_TIME_EVENTS);
    // ［ファイル名を指定して実行］を選択します。
    AIT_PlayKey("r");
    AIT_Sleep(SLEEP_TIME_EVENTS);
    AITEVENTFLAG1 = 1;
    AITIGNORE = 1;
    iLoopCount = 0;
  endif;
  if((AITEVENTFLAG1==1) && (AITIGNORE == 0) && (AIT_FocusWindow("ファイル名を指
定して実行", "#32770") != 0))
    // レジストリファイルのパスを入力します。
    AIT_GetCurrentDirectory(strPath);
    AIT_PlayKey("regedit.exe /s '"" + strPath + "'¥Sample.reg'"");
    AIT_Sleep(SLEEP_TIME_EVENTS);
    // レジストリファイルをインポートします。
```

```
    AIT_PlayKey("{ENTER}");
    AIT_Sleep(SLEEP_TIME_EVENTS);
    iLoopCount = iLoopMax;
    DM_RTN = OK_END;
    continue;
  endif;
  AITIGNORE = 0;
  iLoopCount = iLoopCount + 1;
  AIT_Sleep(SLEEP_TIME);
loop;
```

3. **配布するソフトウェアと同じディレクトリに，作成したレジストリファイルを格納する。**
4. **作成した** AIT **ファイルを指定してパッケージングを実行する。**
5. **リモートインストールを実行する。**
   リモートインストール先のクライアントで，setup.exe などと同じディレクトリにレジストリファイルが展開され，AIT ファイルが実行されます。

5 トラブルシューティング

この章では，Automatic Installation Tool でトラブルが発生した場合の対処方法，およびメッセージについて説明します。

5.1 メッセージの確認

AIT ファイルの実行時やコード解析時に，GUI の操作誤り，ディスク使用超過，メモリ破損などのトラブルが発生した場合は，まず，標準出力またはログファイルにエラーメッセージが出力されているかどうかを確認してください。エラーメッセージが出力されている場合，エラーメッセージのメッセージ ID から，エラーが発生しているプログラムやトラブルの要因が特定できます。

メッセージは，ダイアログボックスまたはアウトプットウィンドウに表示されます。なお，一部のメッセージは**ログファイル**に出力されます。ログファイルの格納先を次に示します。

JP1/NETM/DM のインストール先ディレクトリ ¥LOG

メッセージが出力されるログファイルと，その内容は次のとおりです。

ファイル名	最大行数	内容
ait.log	2000	GUI 関連のエラーがこのファイルに出力されます。
aitapi.log	2000	API 関連のエラーがこのファイルに出力されます。
aitexec.log	2000	実行，解析関連のエラーがこのファイルに出力されます。

5.2 メッセージの形式

このマニュアルでは，メッセージを次の形式で説明します。

メッセージ ID

メッセージテキスト

要因

aa・・・aa

対処

bb・・・bb

例

cc・・・cc

(1) メッセージ ID

メッセージの形式：AIT*Xnnnn-Z* メッセージテキスト

AIT

メッセージを表示したプログラムが Automatic Installation Tool であることを表します。

X

メッセージを出力したコンポーネントを表します。

G：GUI

CE または CW：AIT ファイルの実行および解析時（E は ERROR を，W は WARNING を示します）

nnn **または** *nnnn*

各コンポーネントで任意に割り当てられるメッセージ番号を表します。

Z

メッセージの種別を表します。各種別を説明します。ただし，AITCE，AITCW のメッセージの場合，AITCE，AITCW の「E」および「W」に該当します。

E（ERROR）

致命的なエラーが発生したことを示し，通常は処理が停止するエラーメッセージです。

W（WARNING）

期待された情報が検出されなかったことを示す警告メッセージです。デフォルト値があります。

Q（QUESTION）

ユーザが応答する必要のあるメッセージです。

I（INFORMATION）

重要な活動が実行されていることを示す情報メッセージです。

(2) メッセージテキスト

メッセージの内容を示したテキストです。メッセージ中の <*XXX*> の部分は，該当する文字列が実際には表示されます。

(3) 要因

メッセージが表示された要因の説明です。

(4) 対処

メッセージが表示された場合の対処方法の説明です。

(5) 例

API の誤った使用方法および正しい使用方法の説明です。

5.3 編集ウィンドウ使用時のメッセージ

ここでは，編集ウィンドウ使用時に表示されるメッセージを示します。これらのメッセージは，ダイアログボックスまたはアウトプットウィンドウに表示されます。

AITG100-E

レジストリが無効または壊れている可能性があります。

要因

以下の要因が考えられます。

- 構成レジストリデータベースが壊れています。
- 構成レジストリキーが無効です。
- レジストリが壊れています。レジストリデータを含む一部のファイル構造体が壊れている，システム上のファイルのメモリイメージが壊れている，または代替のコピーかログが存在しないか壊れているためファイルを修復できません。
- システムはレジストリ内にファイルをロードするか復元することを試みましたが，指定されたファイルの形式はレジストリファイル形式ではありませんでした。

対処

該当する追加情報に基づいて，必要な処理を実行してください。レジストリが壊れている場合は，修複ディスクを使用して設定情報を復元してください。
問題が解決しない場合は，システム管理者に連絡してください。

AITG101-E

レジストリの操作が失敗しました。

要因

以下の要因が考えられます。

- 構成レジストリキーを開けません。
- 構成レジストリキーを読み込めません。
- 構成レジストリキーに書き込めません。
- レジストリで修復できない失敗を持つ I/O 操作を開始したため，システムのレジストリイメージを含むファイルの一つに対して，読み取り，書き込み，またはフラッシュを実行できませんでした。
- 「削除」としてすでに指示されているレジストリキーに対し，誤った操作が実行されました。
- レジストリログ内に必要な領域を割り当てられませんでした。
- すでにサブキーまたは値のあるレジストリキーにシンボリックリンクを作成できません。
- 揮発性親キーの下に安定したサブキーを作成しようとしました。
- 対象となるマルチバイトコードページに Unicode 文字のマッピングが存在しません。
- インデックス内に指定されたキーと一致するものがありませんでした。

対処

該当する追加情報に基づいて，必要な処理を実行してください。
問題が解決しない場合は，システム管理者に連絡してください。

AITG102-E

内部でエラーが発生しました。

要因

アプリケーションが以下に示す誤ったハンドルで，リソースにアクセスしました。

- 無効なハンドル
- 無効なウィンドウハンドル

- 無効なフックハンドル

対処

アプリケーションを再実行してください。
問題が解決しない場合は，システム管理者に連絡してください。

AITG103-E

メモリまたはディスクの残容量がありません。

要因

以下の要因が考えられます。

- このコマンドを処理するための十分な記憶容量がありません。
- この処理を完了するための十分な記憶容量がありません。
- ディスクがいっぱいです。
- ディスク上に十分なスペースがありません。
- 要求されたサービスを完了するためにはシステムリソースが不足しています。

対処

使用していないアプリケーションをすべて終了して再度処理を試みてください。
ディスク上の不要なファイルを削除してアプリケーションを再実行してください。
問題が解決しない場合は，システム管理者に連絡してください。

AITG104-E

ファイルまたはディレクトリの操作が失敗しました。

要因

以下の要因が考えられます。

- システムが指定されたファイルを検索できません。
- システムが指定されたパスを検索できません。
- システムがファイルを開けません。
- システムが指定されたドライブを検索できません。
- ディレクトリを削除できません。
- システムがファイルをほかのディスクドライブに移動できません。
- ファイルが存在しません。
- ディレクトリまたはファイルを作成できません。
- ファイル名が長過ぎます。
- ディレクトリがルートディレクトリのサブディレクトリではありません。
- ディレクトリが空ではありません。
- ハードディスクへのアクセス中に，ディスク操作に失敗し，さらに再試行も失敗しました。

対処

該当する追加情報に基づいて，必要な処理を実行してください。
例えば，正しいファイル名かパス名または正しいドライブ名を使用して再試行してください。
問題が解決しない場合は，システム管理者に連絡してください。

AITG105-E

レコーダの起動に失敗しました。

要因

編集ウィンドウから以下のどれかが初期化されています。

- 文法チェック
- 実行

- レコーダ
- ウィンドウプロパティ

対処

アプリケーションを再実行してください。
問題が解決しない場合は，システム管理者に連絡してください。

AITG106-E

デバッグ操作は失敗しました。

要因

デバッグ時にアプリケーションが実行エンジンと交信できません。

対処

アプリケーションを再実行してください。
問題が解決しない場合は，システム管理者に連絡してください。

AITG107-W

シンタクスファイルは見つかりませんでした。シンタクスハイライトは活性化されませんでした。

要因

編集ウィンドウから文法ファイル（AIT.syn）を検索できません。

対処

次のレジストリキーに指定された文法ファイルのパスが，正しいロケーションを指していません。

- OS が 32 ビット版の場合

  ```
  HKEY_LOCAL_MACHINE¥SOFTWARE¥HITACHI¥NETM AIT¥ConfPath
  ```

- OS が 64 ビット版の場合

  ```
  HKEY_LOCAL_MACHINE¥SOFTWARE¥Wow6432Node¥Hitachi¥NETM AIT¥ConfPath
  ```

レジストリ内のパスを検証して，アプリケーションを再実行してください。

AITG108-E

無効な行数です。

要因

［ジャンプ］ダイアログボックスまたは［ブレークポイントの設定］ダイアログボックスの操作時，行番号エディットボックスに無効な行番号が指定されています。

対処

有効な番号を指定して再試行してください。

AITG109-E

文字列 '< 文字列 >' は見つかりません。

要因

［検索］ダイアログボックスまたは［置換］ダイアログボックスの操作時に，アプリケーションが編集ウィンドウのテキストから文字列を検索できません。

対処

必要ありません。

AITG110-E

PACKAGE_INFO **セクションの形式は無効です。**

要因

［パッケージ情報］ダイアログボックスを呼び出す際の，AIT ファイル内の PACKAGE_INFO セクションの形式に誤りがあります。

対処

AIT ファイルの文法をチェックしてから［パッケージ情報］ダイアログボックスを呼び出してください。

AITG111-E

必須項目 '< 項目名 >' を入力してください。

要因

［パッケージ情報］ダイアログボックスで必ず指定する，以下のパッケージ項目が指定されていません。

- パッケージ識別 ID
- パッケージ名
- バージョン
- インストールプログラム名
- インストール先ドライブ
- インストール先ディレクトリ

対処

必ず指定する項目を指定してからパッケージ情報を生成してください。

AITG112-Q

パッケージ情報を更新しますか？

要因

［レコーダ］ダイアログボックスの［停止］ボタンをクリックすると表示されます。AIT ファイル内に PACKAGE_INFO セクションを生成するかどうかを選択します。

対処

PACKAGE_INFO セクションを生成するために［パッケージ情報］ダイアログボックスを表示する場合は，［はい］ボタンをクリックしてください。

PACKAGE_INFO セクションのない AIT ファイルを作成する場合は，［いいえ］ボタンをクリックしてください。

AITG113-E

1 から 64 までの範囲で指定してください。

要因

［オプション］ダイアログボックスの［タブ］パネルの「タブサイズ」に，1 ～ 64 以外の整数値が指定されています。

対処

1 ～ 64 の整数値を指定して再試行してください。

AITG114-E

無効な文字が入力されました。< 指定できる文字列 > を指定してください。

要因

［パッケージ情報］ダイアログボックスの，以下のパッケージ項目のどれかに無効な文字が指定されて

います。
- パッケージ識別 ID
- パッケージ名
- バージョン
- インストールプログラム名
- インストール先ディレクトリ

対処

＜指定できる文字列＞に表示された内容に従って，指定できる文字列を入力してからパッケージ情報を生成してください。

AITG115-W

レジストリキーを更新できません。

要因

以下の要因が考えられます。
- 構成レジストリデータベースが壊れています。
- 構成レジストリキーが開けません。
- 構成レジストリキーが読めません。
- 構成レジストリキーを更新できません。
- 構成レジストリキーが無効です。
- レジストリが壊れています。レジストリデータを含む一部のファイル構造体が壊れている，システム上のファイルのメモリイメージが壊れている，または代替のコピーかログが存在しないか壊れているため，ファイルを修復できません。
- システムはレジストリ内のファイルの更新を試みましたが，指定されたファイルの形式はレジストリファイル形式ではありません。

対処

該当する追加情報に基づいて，必要な処理を実行してください。

AITG116-W

レジストリキーを読み込めません。

要因

以下の要因が考えられます。
- 構成レジストリデータベースが壊れています。
- 構成レジストリキーが開けません。
- 構成レジストリキーが読めません。
- 構成レジストリキーが無効です。
- レジストリが壊れています。レジストリデータを含む一部のファイル構造体が壊れている，システム上のファイルのメモリイメージが壊れている，または代替のコピーかログが存在しないか壊れているため，ファイルを修復できません。
- システムはレジストリからのファイルの読み出しを試みましたが，指定されたファイルの形式はレジストリファイル形式ではありません。

対処

該当する追加情報に基づいて，必要な処理を実行してください。

AITG117-W

レジストリキーを削除できません。

要因

以下の要因が考えられます。
- 構成レジストリデータベースが壊れています。
- 構成レジストリキーを削除できません。
- 構成レジストリキーが無効です。
- レジストリが壊れています。レジストリデータを含む一部のファイル構造体が壊れている，システム上のファイルのメモリイメージが壊れている，または代替のコピーかログが存在しないか壊れているため，ファイルを修復できません。
- システムはレジストリからのファイルの削除を試みましたが，指定されたファイルの形式はレジストリファイル形式ではありません。

対処

該当する追加情報に基づいて，必要な処理を実行してください。

AITG118-W

AIT エンジンの処理が実行されています。ファイルを閉じることはできません。

要因

文法チェック中に，処理中の AIT ファイルを閉じる操作が実行されています。

対処

必要ありません。

AITG119-W

AIT エンジンの処理が実行されています。アプリケーションを閉じることはできません。

要因

以下の要因が考えられます。
- AIT ファイルが文法チェック中に，アプリケーションを閉じようとした場合
- 実行エンジンによって AIT ファイルが実行されている際に，アプリケーションを閉じようとした場合

対処

必要ありません。

AITG120-E

シンボルが見つかりませんでした。

要因

以下の要因が考えられます。
- デバッグモード時に AIT ファイルで定義されていない変数を監視ウィンドウに入力した場合
- デバッグモード時に AIT ファイルで定義されていない記号を入力した場合

対処

監視ウィンドウに入力する記号が，デバッグ中の AIT ファイルで有効な変数であることを確認してください。

AITG121-W

オプションを更新する権限がありません。保存されたオプションは現在のセッションだけに反映されます。

要因

ゲストとしてログオンしているときに，レジストリの値が保存されました。

対処

必要ありません。

AITG122-E

< 各パッケージ項目に入力できる最大文字数 > 文字を超えています。

要因

PACKAGE_INFO セクションの各パッケージ項目に入力できる最大文字数を超えた状態で，PACKAGE_INFO セクションが生成されています。

対処

各パッケージ項目は，最大文字数以下の文字数で入力してください。

AITG123-E

< 各パッケージ項目に入力できる最大文字数 > 文字を超えています。

要因

PP 識別情報ファイルの各パッケージ項目に入力できる最大文字数を超えた状態で，PP 識別情報ファイルが生成されています。

対処

各パッケージ項目で指定できる文字数を確認して，指定できる範囲内の文字数で入力してください。

AITG124-E

PP 識別情報ファイルの操作に失敗しました。

要因

ほかのアプリケーションで操作されている可能性があります。

対処

ほかのアプリケーションで操作されている PP 識別情報ファイルを閉じてください。

AITG125-E

項目 '< 項目名 >' を入力してください。

要因

［パッケージ情報］ダイアログボックスで，次の項目のどちらかが指定されていません。

- 「AIT ファイルの格納パス」
- 「識別用ファイル名」

対処

項目を指定してからパッケージ情報を生成してください。

AITG200-E

レコーディングの操作が失敗しました。

要因

アプリケーションが以下に示す誤ったハンドルでリソースにアクセスしました。

- 無効なハンドル
- 無効なウィンドウハンドル
- 無効なフックハンドル

対処

アプリケーションを再実行してください。

問題が解決しない場合は，システム管理者に連絡してください。

AITG201-E

ウィンドウ詳細の取得に失敗しました。

要因

レコーディング時に，ウィンドウの情報を取得できませんでした。

対処

問題が解決しない場合は，システム管理者に連絡してください。

AITG202-E

コントロール詳細の取得に失敗しました。

要因

レコーディング時に，ウィンドウのコントロール情報を取得できませんでした。

対処

問題が解決しない場合は，システム管理者に連絡してください。

AITG203-E

イベント詳細の取得に失敗しました。

要因

レコーディング時に，イベント情報を取得できませんでした。

対処

問題が解決しない場合は，システム管理者に連絡してください。

AITG204-E

AIT **ファイルで** PACKAGE_INFO **セクションの生成に失敗しました。**

要因

レコーディング終了後に PACKAGE_INFO セクションを生成できませんでした。

対処

一時フォルダを削除して再試行してください。
問題が解決しない場合は，システム管理者に連絡してください。

AITG205-E

AIT **ファイルで** DEFINE **セクションの生成に失敗しました。**

要因

レコーディング終了後に DEFINE セクションを生成できませんでした。

対処

一時フォルダを削除して再試行してください。
問題が解決しない場合は，システム管理者に連絡してください。

AITG206-E

AIT **ファイルで** MAIN **セクションの生成に失敗しました。**

要因

レコーディング終了後に MAIN セクションが生成できませんでした。

対処

一時フォルダを削除して再試行してください。
問題が解決しない場合は，システム管理者に連絡してください。

AITG208-E

AIT ファイルの生成に失敗しました。

要因

レコーディング終了後に AIT ファイルを生成できませんでした。
以下の要因が考えられます。

- ハードディスクへのアクセス中に，ディスク操作に失敗し，さらに再試行も失敗しました。
- このコマンドを処理するための十分な記憶容量がありません。
- この処理を完了するための十分な記憶容量がありません。
- 無効なハンドルです。

対処

該当する追加情報に基づいて，必要な処理を実行してください。
問題が解決しない場合は，システム管理者に連絡してください。

AITG209-I

インストールプログラム < インストールプログラム名 > の実行は成功しました。

要因

AIT ファイルが正常に実行されました。

対処

必要ありません。

AITG211-W

AIT ファイルのためのログオプションは，レジストリに設定されませんでした。

要因

レジストリ内にログオプションが設定されていません。

対処

レジストリ内のログオプションを設定してください。

AITG212-I

レコーディングの処理が完了しました。

要因

レコーディングが正常に完了しました。

対処

必要ありません。

AITG213-I

AIT ファイルの作成が成功しました。

要因

レコーディング終了後に AIT ファイルが正常に生成されました。

対処

必要ありません。

AITG214-W

ユーザ情報ファイルの更新は，存在しないイベントのためにスキップしました。

要因

一時停止後にレコーダがレコーディングを続行できません。

対処

必要ありません。

AITG215-E

レコーディングを継続できませんでした。

要因

一時停止後にレコーダがレコーディングを続行できません。

対処

該当する追加情報に基づいて，必要な処理を実行してください。

AITG216-Q

インストールプログラムの起動に失敗しました。: **< インストールプログラム名 > < 追加情報 > レコーディングを続けますか。**

要因

［レコーダ］ダイアログボックスで，誤ったインストールプログラム名が指定されています。
以下の要因が < 追加情報 > として表示されます。

- 指定されたファイルを見つけることができません
- 指定されたパスを見つけることができません
- アクセスが拒否されました
- メモリが不足しているため処理を行えません

対処

レコーディングを開始する場合は［はい］ボタンを，追加情報に従って必要な処置を実行する場合は［いいえ］ボタンをクリックしてください。

5.4 実行および解析時のメッセージ

ここでは，AIT ファイル解析時および実行時に表示されるメッセージを示します。これらのメッセージは，アウトプットウィンドウに表示されます。

AITCE-0001

< トークン 1> は予期しないものです。< トークン 2> は見つかりません。

要因

無効なキーワードまたはシンボル < トークン 1> が検出されました。スクリプト解析では，キーワードまたはシンボル < トークン 2> を予期しています。

対処

必要に応じて < トークン 2> を追加してください。

例

誤った指定例

```
DEFINE
{
   const integer OK_END = 0;
   const integer NG_END = -1      // 「;」が指定されていません
   const integer sloop_max = 30;
}
```

正しい指定例

```
DEFINE
{
   const integer OK_END = 0;
   const integer NG_END = -1;      // 「;」を追加しました
   const integer sloop_max = 30;
}
```

AITCE-0002

< トークン > は予期しないものです。

要因

無効なキーワードまたはシンボル < トークン > が検出されました。

対処

予期されないキーワードまたはシンボルを削除してください。

例

誤った指定例

```
DEFINE
{
   integer OK_END = 0;
   integer sloop_max = 0;;      // 「;」が二つ指定されています
}
```

変数「sloop_max」に「;」が二つ指定されています。

正しい指定例

```
DEFINE
{
   integer OK_END = 0;
   integer sloop_max = 0;      // 「;」を削除しました
}
```

変数「sloop_max」の「;」を一つ削除しました。

AITCE-0003

関数名を識別子名として使用しています。

要因

関数名は識別子として使用できません。

対処

識別子名を変更してください。

例

誤った指定例

```
DEFINE
{
   integer AIT_LogMessage = 10;
}
```

変数名に関数名が使用されています。

正しい指定例

```
DEFINE
{
   integer AIT_LogMessageNumber = 10;
}
```

変数名を「AIT_LogMessageNumber」に変更しました。

AITCE-0005

階層数が 255 を超えています。

要因

AIT ファイル内で構造体の階層レベルが最大値の 255 を超えています。

対処

if，if-else，do-while，while または switch 構造体の階層レベルが最大値の 255 を超えると，AIT ファイルの解析ができなくなります。AIT ファイルを再編集し，構造体の階層レベルのネストを浅くしてください。

AITCE-0006

識別子名が 64 文字を超えています。

要因

各識別子名に使用できる最大文字数は 64 です。この最大文字数を超える識別子名が AIT ファイルに存在します。

対処

識別子名は 1 ～ 64 文字で指定してください。

例

誤った指定例

```
DEFINE
{
   integer sloop_max = 0;
   integer
sample123456sample123456sample123456sample123456sample123456sample = 0;
// 66文字で変数名が指定されています
}
```

2 番目に定義した変数名が最大文字数を超えています。

正しい指定例

```
DEFINE
{
   integer sloop_max = 0;
   integer sample123456 = 0;   // 64文字以内で変数名を指定しました
}
```

2 番目に定義した変数名を最大文字数内で指定しました。

AITCE-0007

文字列定数が複数列にわたっています。

要因

最初の行が ¥n で終わっているため，文字列定数が 2 行目に続いています。

対処

文字列定数は 1 行で指定してください。

例

誤った指定例

```
DEFINE
{
   integer sloop_max = 0;
   string SoftwareName = " My
                                 Setup";      // 文字列定数が2行目に続いています
}
```

変数「SoftwareName」の文字列定数が 2 行目に続いています。

正しい指定例

```
DEFINE
{
   integer sloop_max = 0;
   string SoftwareName = " My Setup";   // 文字列定数を1行で指定しました
}
```

変数「SoftwareName」の文字列定数を 1 行で指定しました。

AITCE-0008

エスケープシーケンスの使用方法が誤っています。

要因

実行時に指定された AIT ファイルの文字列定数に，誤ったエスケープシーケンスが含まれています。

対処

文字列定数には，正しいエスケープシーケンスを使用してください。

例

誤った指定例

```
DEFINE
{
   integer sloop_max = 0;
   string str1 = "sample'
                 testing";
}
```

文字列変数「strl」は「'」付きの文字列が代入されていますが，その文字列が複数行にわたって記述されています。
文字列が連続していることを示す文字「_」を「'」の代わりに使用する必要があります。

正しい指定例

```
DEFINE
```

```
{
   integer sloop_max=10;
   string str1 = "sample_
                  testing";
}
```

「'」を削除してエスケープシーケンスの誤りを修正しました。

AITCE-0009

case 内で識別子は使用できません。

要因

case 内で識別子を指定しています。

対処

case 内では，定数またはマクロだけを使用してください。

例

誤った指定例

```
DEFINE
{
   const string FileVersion = "7.1";
   string stMsgText;
}
MAIN
{
   switch (FileVersion)
      case stMsgText:      // caseに識別子が指定されています
         ...
         break;
      default:
         ...
         break;
   endswitch;
}
```

case 内に，識別子「stMsgText」が指定されています。

正しい指定例

```
DEFINE
{
   const string FileVersion = "7.1";
   string stMsgText;
}
MAIN
{
   switch (FileVersion)
      case "7.1":      // 識別子の代わりに文字列定数を指定しました
         ...
         break;
      default:
         ...
         break;
   endswitch;
}
```

case 内に，識別子「stMsgText」の代わりに文字列定数を指定しました。

AITCE-0010

"+" または "-" の使用方法が誤っています。

要因

AIT ファイルで，「++」，「--」，「+-」，または「-+」演算子が指定されています。AIT ファイルの解析では，これらの演算子を解析できません。

対処

「++」,「--」,「+-」, または「-+」演算子を AIT ファイルから削除してください。

例

誤った指定例

```
DEFINE
{
   integer sloop_count = 0;
}
MAIN
{
   if (AIT_FileExists("#setup.exe") == 0)
      ...
      ...
      sloop_count++;      // 誤った演算子「++」が変数の増分に使用されています
   endif;
}
```

インクリメント演算子とディクリメント演算子は使用できません。
変数「sloop_count」が「++」で増分されているため, エラーメッセージが表示されます。

正しい指定例

```
DEFINE
{
   integer sloop_count = 0;
}
MAIN
{
   if (AIT_FileExists("#setup.exe") == 0)
      ...
      ...
      sloop_count = sloop_count + 1;      // 「++」を削除しました
   endif;
}
```

「++」を変数「sloop_count」から削除しました。

AITCE-0011

AIT ファイルが 65535 行を超えています。

要因

AIT ファイル内の最大行数は 65,535 です。AIT ファイルがこの最大値を超えると, AIT ファイルの解析を停止します。

対処

AIT ファイル内の行数を減らしてください。

AITCE-0012

< 識別子 > : 定義されていない識別子です。

要因

変数の型を定義しないで変数が指定されています。

対処

DEFINE セクションで変数を定義してください。

例

誤った指定例

```
DEFINE
{
   string stMsgText;
}
```

```
MAIN
{
   if (AIT_FileExists("#setup.exe") == 0)
      stMsgText = "Setup(Japanese) " + InstallerName + " Not Found";
      AIT_LogMessage(stMsgText);
      sloop_max = 0;      // この変数はDEFINEセクションで定義されていません
   endif;
}
```

DEFINE セクションで変数「sloop_max」が定義されていないまま，MAIN セクションでこの変数が指定されています。

正しい指定例

```
DEFINE
{
   string  stMsgText;
   integer sloop_max;      // 変数「sloop_max」を定義しました

}
MAIN
{
   if (AIT_FileExists("#setup.exe") == 0)
      stMsgText = "Setup(Japanese) " + InstallerName + " Not Found";
      AIT_LogMessage(stMsgText);
      sloop_max = 0;      // 定義した変数を利用しました
   endif;
}
```

変数「sloop_max」が DEFINE セクションで定義され，その変数が MAIN セクションで指定されました。

AITCE-0013

< 識別子 > : 再定義されました。

要因

すでに識別子が定義されている場合に，それと同じ識別子が再定義されています。

対処

再定義されている変数を削除してください。

例

誤った指定例

```
DEFINE
{
   integer sloop_max = 10;
   string sloop_max = "sample";      // 変数「sloop_max」がstring型として再定義されています。
}
```

すでに integer 型として定義されている変数「sloop_max」が，string 型変数として再定義されています。

正しい指定例

```
DEFINE
{
   integer sloop_max = 10;
   string stMsgText = "sample";      // 変数名を変更しました
}
```

変数「sloop_max」の再定義を削除し，異なる変数名を定義しました。

AITCE-0014

パッケージ情報のフィールド < パッケージ項目 > のデータ < 値 > が誤っています。

要因

PACKAGE_INFO セクション内のパッケージ項目のどれかに無効な値が含まれています。

対処

パッケージ情報のフィールドに適した値を指定してください。

例

誤った指定例

```
PACKAGE_INFO
{
   PackageID = "#$@#ADOBEACROBATREADER";      // PackageIDに無効な文字#$@#
が含まれています
   Product = "Adobe Acrobat Reader 5.05";
   Version = "505";
   InstallerName = "Ar505jpn.exe";
   InstallDrive = "C:";
   InstallDirectory = "'¥Program Files'¥Adobe";
}
```

PACKAGE_INFO セクションの PackageID に，使用できない文字列「#$@」が含まれています。

正しい指定例

```
PACKAGE_INFO
{
   PackageID = "ACROBAT-READER";   // 使用できない文字 #$@#を削除しました
   Product = "Adobe Acrobat Reader 5.05";
   Version = "505";
   InstallerName = "Ar505jpn.exe";
   InstallDrive = "C:";
   InstallDirectory = "'¥Program Files'¥Adobe";
}
```

PACKAGE_INFO セクションの PackageID から，使用できない文字列「#$@」を削除しました。

AITCE-0015

左辺値は const **オブジェクトに指定されています。**

要因

constant 型として定義された変数に MAIN セクションで値が割り当てられています。

対処

constant 定義を変更するか，割り当てた左辺を削除してください。

例

誤った指定例

```
DEFINE
{
   const float SLEEP_TIME = 2.0;
   integer sloop_count;
}
MAIN
{
   if (AIT_FocusWindow("Setup", "#32770",0.0) > 0)
      AIT_PlayKey("{Enter}");
      AIT_LogMessage("Setup: Enter");
      sloop_count = 0;
      SLEEP_TIME = 3.0;      // 変数「SLEEP_TIME」のconstant値を変更しようとし
ています
      AIT_Sleep(SLEEP_TIME);
   endif;
}
```

変数「SLEEP_TIME」は，constant float 型として定義されていますが，MAIN セクション内で別の値が割り当てられました。

正しい指定例

```
DEFINE
{
   const float SLEEP_TIME = 3.0;
   integer sloop_cnt;
}

MAIN
{
   if (AIT_FocusWindow("Setup", "#32770",0.0) > 0)
      AIT_PlayKey("{Enter}");
      AIT_LogMessage("Setup : Enter");
      sloop_cnt = 0;
      AIT_Sleep(SLEEP_TIME);
   endif;
}
```

MAIN セクション内で，変数「SLEEP_TIME」の constant float 値への割り当てを削除しました。

AITCE-0018

AIT ファイルの構文誤りによって，AIT ファイルの解析が異常終了しました。

要因

文法チェック中に致命的なエラー（文字列の終了処理が正しくない，変数名内の文字数が正しく指定されていない）が発生しています。

対処

エラー個所を修正したあと，AIT ファイルを再度確認してください。

例

誤った指定例

```
DEFINE
{
   string ErrorTxt = "ABC
def";
}
```

文字列定数は，複数の行にわたって指定できませんが，文字列が複数の行にわたって指定されています。

正しい指定例

```
DEFINE
{
   string ErrorTxt = "ABC_
def";      // 「_」で文字列が連続していることを示しています。
}
```

文字列の連続を示す「_」を付けて指定しました。

AITCE-0019

除算の 2 番目のオペランドは 0 です。

要因

AIT ファイルの解析時に除算の 2 番目のオペランドが 0 のため，未定義として扱われています。

対処

0 以外の値を除数として指定してください。

例

誤った指定例

```
DEFINE
```

```
{
   integer sloop_count = 0;
   const integer  sloop_max = 30;
}
MAIN
{
   sloop_count = sloop_max / 0;      // 除数が0であるため，エラーになります
}
```

0 による除算はできませんが，変数「sloop_cnt」を 0 で除算しています。

正しい指定例

```
DEFINE
{
   integer sloop_count = 0;
   const integer sloop_max = 30;
}
MAIN
{
   sloop_count = sloop_max / 1;      // 除数が0以外の値に変更しました
}
```

0 による変数「sloop_max」の除算を削除しました。

AITCE-0020

< データ型 1> と < データ型 2> は < 処理名 > と互換性がありません。

要因

演算子の左辺値と右辺値で互換性のないデータ型が指定されています。例えば，string 型と integer 型が比較に使用された場合，このメッセージが表示されます。

対処

演算子の左辺値と右辺値の両方で互換性のあるデータ型を指定してください。

例

誤った指定例

```
DEFINE
{
   const integer ExeVersion = 7;
   const string FileVersion = "7";
}
MAIN
{
   if (ExeVersion == FileVersion)   // string型とinteger型が比較に使用されて
います
      ...
      ...
   endif;
}
```

integer 型変数「ExeVersion」と string 型変数「FileVersion」が比較に使用されています。string 型と integer 型には，比較演算のための互換性がありません。

正しい指定例

```
DEFINE
{
   const integer ExeVersion = 7;
   const integer FileVersion = 7;
}
MAIN
{
   if (ExeVersion == FileVersion)      // 同じ型を比較に使用しました
      ...
      ...
   endif;
}
```

string 型変数「FileVersion」を integer 型変数に変更し，integer 型変数との比較に変更しました。

AITCE-0021

オペレーション < 演算子名 > で < データ型 1> の使用方法が誤っています。

要因

演算で不適切なデータ型が指定されています。例えば，float 型が剰余（%）演算に使用された場合や，string 型が ! 演算に使用された場合，このメッセージが表示されます。

対処

各演算で使用できるデータ型を指定してください。

例

誤った指定例

```
DEFINE
{
   float SLEEP_TIME = 7.1;
}
MAIN
{
   SLEEP_TIME = SLEEP_TIME % 2;       // floatは剰余演算で使用できないため，エラーになります
}
```

float 型は剰余演算に使用できませんが，float 型変数「SLEEP_TIME」が剰余演算に使用されています。

正しい指定例

```
DEFINE
{
   integer SLEEP_TIME = 7;
}
MAIN
{
   SLEEP_TIME = SLEEP_TIME % 2;       // integer型変数を剰余演算に使用しました
}
```

integer 値を剰余演算に使用しました。

AITCE-0022

integer **値は** '-2147483648' **〜** '2147483647' **の範囲で指定してください。**

要因

許容範囲外の integer 値が指定されています。

対処

'-2,147,483,648' 〜 '2,147,483,647' で integer 値を指定してください。

例

誤った指定例

```
DEFINE
{
   const integer OK_END = 21474836476;      // integer値が最大値を超えています
}
```

変数「OK_END」に最大値よりも大きい integer 値が指定されています。

正しい指定例

```
DEFINE
```

```
{
   const integer OK_END =214748364;      // integer値の範囲内で指定しました
}
```

変数「OK_END」に integer 値の範囲内の値を指定しました。

AITCE-0023

float 値は '3.402823466e+38' 〜 '1.175494351e-38' の範囲で指定してください。

要因

許容範囲外の float 値が指定されています。

対処

'3.402823466e+38' 〜 '1.175494351e-38' で float 値を指定してください。

例

誤った指定例

```
DEFINE
{
   const float NG_END = 3.402823466e+40;      // float値が最大値を超えています
}
```

変数「NG_END」に最大値よりも大きい float 値が指定されています。

正しい指定例

```
DEFINE
{
   const float NG_END = 3.402823466e+10;      // float値の範囲内で指定しました
}
```

変数「NG_END」に float 値の範囲内の値を指定しました。

AITCE-0024

switch 式の求める数値のデータ型が誤っています。

要因

case ラベル値のデータ型が switch ステートメントの case のデータ型と一致していません。例えば，switch ステートメントが integer 型の場合に case 値に float 値が含まれていると，このメッセージが表示されます。

対処

switch ステートメントと case ラベル値で一致するデータ型を指定してください。

例

誤った指定例

```
DEFINE
{
   const string FileVersion = "7.1";
   integer sloop_max = 0;
}
MAIN
{
   switch (FileVersion)   // switchステートメントはstring型です
       case 7.1:      // caseにinteger値が指定されています
         ...
         ...
         break;
      default:
         ...
         ...
         break;
   endswitch;
```

```
}
```

switch ステートメントのデータ型は string 型ですが，case 値は integer 型になっています。

正しい指定例

```
DEFINE
{
   const string FileVersion = "7.1";
   integer sloop_max = 0;
}
MAIN
{
   switch (FileVersion)
      case "7.1":    // caseラベルにstring値を指定しました
         ...
         ...
         break;
      default:
         ...
         ...
         break;
   endswitch;
}
```

case 値に string 型を指定し，switch 文のデータ型と一致するよう変更しました。

AITCE-0025

コメントで予期しない EOF **が検出されました。**

要因

「/*」で開始されたコメントがファイルの終わりまでに「*/」で終了されていません。

対処

ファイルの終わりまでに，コメントを閉じる「*/」を指定してください。

例

誤った指定例

```
DEFINE
{
   /* AITファイルで使用するデータ型    // コメントが閉じられていません
   const integer NG_END = -1
   const integer sloop_max = 30;
}
```

コメントはすべて閉じる必要があります。この例では，「/*」でコメントが開始されていますが，「*/」で終了されていません。

正しい指定例

```
DEFINE
{
   /* AITファイルで使用するデータ型 */   // コメントが終了されています
   const integer NG_END = -1;
   const integer sloop_max = 30;
}
```

コメントが「*/」で終了されました。

AITCE-0026

< 文字コード（16 **進数）> は不正な文字です。**

要因

AIT 言語仕様で規定されていない文字が存在します。文字コードが 16 進数で表示されます。

対処

AIT 言語仕様に記載されている有効な文字だけを指定してください。

例

誤った指定例

```
DEFINE
{
   const integer OK_END = 0;
   const integer NG_END = -1;
   const integer sloop_max = #30;       // 無効な文字「#」があります
}
```

「#$@」といった無効な文字は，文字列定数内で使用できませんが，「#」が変数「sloop_max」に含まれています。

正しい指定例

```
DEFINE
{
   const integer OK_END =0;
   const integer NG_END = -1;
   const integer sloop_max = 30;     // 無効な文字「#」を削除しました
}
```

「#」を変数「sloop_max」から削除しました。

AITCE-0027

式の中では void 型を使用できません。有効な型を使用してください。

要因

void を返す関数が式で使用されています。

対処

void を返す関数は式で使用しないでください。

AITCE-0029

case の値 <case ラベル値 > はすでに使われています。

要因

switch ステートメントで同じ case 値が複数回使用されています。

対処

固有の case ラベル値を指定してください。

例

誤った指定例

```
DEFINE
{
   const string FileVersion = "7.1";
   string stMsgText;
}
MAIN
{
   switch(FileVersion )
      case "7.1":
         ...
         ...
         break;
      case "7.1":      // case値"7.1"はすでに使用されています
         ...
         ...
         break;
      default:
```

```
            ...
            ...
            break;
      endswitch;
   }
```

switch ステートメント内で固有の case ラベル値を指定する必要があります。
ここで，2 番目の case ラベル値 "7.1" は最初の case ラベル値と同じです。

正しい指定例

```
   DEFINE
   {
      const string FileVersion = "7.1";
      string stMsgText;
   }
   MAIN
   {
      switch(FileVersion )
         case "7.1":      // 重複するcase値"7.1"を削除しました
            ...
            ...
            break;
         default:
            ...
            ...
            break;
      endswitch;
   }
```

2 番目の case ラベル値を削除しました。

AITCE-0030

< 関数名 > : 関数名が誤っています。

要因

無効な関数名（使用できる API 一覧にない関数名）が指定されています。

対処

AIT ファイルで有効な関数名を指定してください。有効な関数（API 名）の一覧は,「4.1　API 一覧」を参照してください。

例

誤った指定例

```
   DEFINE
   {
      const string ExeVersion = "7.1";
      const string FileVersion = "7.1";
      string stMsgText;
   }
   MAIN
   {
      if (ExeVersion == FileVersion)
         if (AIT_FileExists1("#setup.exe") == 0)      // 「AIT_FileExists1」
   は使用できるAPIではありません
            stMsgText = "Setup(Japanese) " + InstallerName + " Not Found";
            AIT_LogMessage(stMsgText);
         endif;
      endif;
   }
```

「AIT_FileExists1()」は使用できる API ではありません。

正しい指定例

```
   DEFINE
   {
      const string ExeVersion = "7.1";
      const string FileVersion = "7.1";
```

```
   string stMsgText;
}
MAIN
{
   if (ExeVersion == FileVersion)
     if (AIT_FileExists("setup.exe") == 0)     // 使用できるAPIを指定しました
         stMsgText = "Setup(Japanese) " + InstallerName + " Not Found";
         AIT_LogMessage(stMsgText);
      endif;
   endif;
}
```

使用できる API として「AIT_FileExists()」を指定しました。

AITCE-0031

< 関数名 > : 引数 < 指定されている引数の数 > を取得できません。

要因

引数の数に誤りのある関数が含まれています。
関数にはそれぞれ独自の引数のセットがあります。関数に対して指定された引数の数が，実際に呼び出させる関数の引数の数と一致しないと，このメッセージが呼び出されます。

対処

関数に対して正しい数の引数を指定してください。実際の引数のリストについては，「4.2　API の詳細」を参照してください。

例

誤った指定例

```
DEFINE
{
   const string ExeVersion = "7.1";
   const string FileVersion = "7.1";
   string stMsgText;
}
MAIN
{
   if (ExeVersion == FileVersion)
      if (AIT_FileExists() == 0)      // 「AIT_FileExists」に引数が指定されていません
         stMsgText = "Setup(Japanese) " + InstallerName + " Not Found";
         AIT_LogMessage(stMsgText);
      endif;
   endif;
}
```

「AIT_FileExists()」に対しては，一つ以上の引数を指定してください。この例では「AIT_FileExists()」に対して引数が指定されていません。

正しい指定例

```
DEFINE
{
   const string ExeVersion = "7.1";
   const string FileVersion = "7.1";
   string stMsgText;
}
MAIN
{
   if (ExeVersion == FileVersion)
      if (AIT_FileExists("#setup.exe") == 0)
      // 「AIT_FileExists」に対して引数を指定しました
         stMsgText = "Setup(Japanese) " + InstallerName + " Not Found";
         AIT_LogMessage(stMsgText);
      endif;
   endif;
}
```

「AIT_FileExists()」に対して引数を指定しました。

AITCE-0032

関数に指定された一つまたは複数の変数のデータ型が誤っています。

要因

引数のデータ型と一致しないデータ型が関数に含まれています。
関数にはそれぞれ独自の引数のセットがあります。関数に対して指定された引数のデータ型が，実際に呼び出させる関数の引数のデータ型と一致しないと，このメッセージが呼び出されます。

対処

関数の引数で正しいデータ型を指定してください。実際の引数のリストについては，「4.2　API の詳細」を参照してください。

例

誤った指定例

```
DEFINE
{
   const string ExeVersion = "7.1";
   const string FileVersion = "7.1";
   string stMsgText;
}
MAIN
{
   if (ExeVersion == FileVersion )
      if (AIT_FileExists("#setup.exe") == 0)
         stMsgText = "Setup(Japanese) " + InstallerName + " Not Found";
         AIT_LogMessage(stMsgText);
         AIT_Sleep(3);      // integer値が「AIT_Sleep」に指定されています
      endif;
   endif;
}
```

「AIT_Sleep()」には，float 型の引数だけが指定できますが，integer 値が引数として指定されています。

正しい指定例

```
DEFINE
{
   const string ExeVersion = "7.1";
   const string  FileVersion = "7.1";
   string stMsgText;
}
MAIN
{
   if (ExeVersion == Version)
      if (AIT_FileExists("#setup.exe") == 0)
         stMsgText = "Setup(Japanese) " + InstallerName + " Not Found";
         AIT_LogMessage(stMsgText);
         AIT_Sleep(3.1);      // integer値をfloat値に変更しました
      endif;
   endif;
}
```

「AIT_Sleep()」の引数を integer から float に変更しました。

AITCE-0034

< ラベル名 > : 定義されていないラベルです。

要因

ラベルを定義しないで goto ステートメントが使用されています。
AIT ファイルでは，各 goto ステートメントに対して関連するラベルを定義する必要があります。

対処

すべての goto ステートメントに対して関連ラベル名を指定してください。

例

誤った指定例

```
DEFINE
{
   integer sloop_max = 0;
}
MAIN
{
   goto ErrorLabel; // ラベル「ErrorLabel」が定義されていません
   sloop_max = 0;
}
```

goto ステートメントに対して，関連するラベルステートメントを指定する必要があります。ここでは，goto ErrorLabel しか指定されていません。ラベル「ErrorLabel」は MAIN セクションのどこにも定義されていません。

正しい指定例

```
DEFINE
{
   integer sloop_max = 0;
}
MAIN
{
   goto ErrorLabel;
   sloop_max = 0;
ErrorLabel:      // ラベルを定義しました
}
```

goto ステートメントに対してラベル「ErrorLabel」を指定しました。

AITCE-0037

goto のラベル < ラベル名 > は別のセクションにあります。

要因

指定されたラベル名とそれに対応する goto ステートメントが別のセクションに存在します。
goto ステートメントとそれに関連するラベルは，同じセクション内にある必要があります。

対処

ラベル名と goto ステートメントは同じセクション内で指定してください。

例

誤った指定例

```
DEFINE
{
   string stMsgText;
}
MAIN
{
   goto ErrorLabel;      // gotoステートメントがMAINセクションで定義されています
}
ERROR
{
ErrorLabel:      // ラベルがERRORセクションで定義されています
      stMsgText = "Setup(Japanese) " + InstallerName + " Not Found";
}
```

ラベルとその goto ステートメントは，同じセクション内で定義する必要があります。この例では，goto ステートメントが MAIN セクション内にあり，ラベル「ErrorLabel」が ERROR セクション内にあります。

正しい指定例

```
DEFINE
{
   string stMsgText;
}
MAIN
{
      goto ErrorLabel;
ErrorLabel:      // ラベルがMAINセクション内で定義されています
      stMsgText = "Setup(Japanese) " + InstallerName + " Not Found";
}
```

ラベル「ErrorLabel」とその goto ステートメントを MAIN セクション内で定義しました。

AITCE-0038

変数の設定値の型が誤っています。

要因

変数に対して指定された値が変数のデータ型と異なります。例えば，string 値が integer 型変数に割り当てられると，このメッセージが呼び出されます。

対処

変数と同じ型の値を指定してください。

例

誤った指定例

```
DEFINE
{
   const integer OK_END =0;
   const integer NG_END = -1;
   const integer sloop_max = "30";      // string値がinteger型変数に割り当てられています
}
```

この例では，文字列定数が整数定数「sloop_max」に割り当てられています。

正しい指定例

```
DEFINE
{
   const integer OK_END =0;
   const integer NG_END = -1;
  const integer sloop_max = 30;     // string値でなくinteger値を割り当てました
}
```

文字列定数を削除し，integer 値の変数「sloop_max」を割り当てました。

AITCE-0039

break の使用方法が誤っています。

要因

do-while，while-loop，for-next，または switch ステートメント以外で break ステートメントが使用されています。break ステートメントはループ構造体内だけで有効です。

対処

break ステートメントは，do-while，while-loop，for-next，または switch ステートメント内で使用してください。

例

誤った指定例

```
DEFINE
{
```

```
   const string ExeVersion = "7.1";
   const string FileVersion = "7.1";
   string stMsgText;
}
MAIN
{
   if (ExeVersion == FileVersion )
      if (AIT_FileExists("#setup.exe") == 0)
         stMsgText = "Setup(Japanese) " + InstallerName + " Not Found";
         AIT_LogMessage(stMsgText);
         break;      // breakステートメントがifステートメント内にあります
      endif;
   endif;
}
```

この例では，break ステートメントが if 構造体内にあります。

正しい指定例

```
DEFINE
{
   const string ExeVersion = "7.1";
   const string  FileVersion = "7.1";
   string stMsgText;
}
MAIN
{
   if (ExeVersion == FileVersion)
      if (AIT_FileExists("#setup.exe") == 0)
         stMsgText = "Setup(Japanese) " + InstallerName + " Not Found";
         AIT_LogMessage(stMsgText);
      endif;
   endif;
}
```

break ステートメントを if 構造体から削除しました。

AITCE-0040

continue の使用方法が誤っています。

要因

do-while，while-loop，または for-next 以外で continue ステートメントが使用されています。

continue ステートメントは，ループ構造体内だけで有効です。

対処

continue ステートメントは，do-while，while-loop，または for-next ステートメント内だけで使用してください。

例

誤った指定例

```
DEFINE
{
   const string ExeVersion = "7.1";
   const string FileVersion = "7.1";
   string stMsgText;
}
MAIN
{
   if (ExeVersion == FileVersion)
      if (AIT_FileExists("#setup.exe") == 0)
         stMsgText = "Setup(Japanese) " + InstallerName + " Not Found";
         AIT_LogMessage(stMsgText);
         continue;      // continueステートメントがifステートメント内にあります
      endif;
   endif;
}
```

この例では，continue が if 構造体内にあります。

正しい指定例

```
DEFINE
{
   const string ExeVersion = "7.1";
   const string  FileVersion = "7.1";
   string stMsgText;
}
MAIN
{
   if (ExeVersion == FileVersion)
      if (AIT_FileExists("#setup.exe") == 0)
         stMsgText = "Setup(Japanese) " + InstallerName + " Not Found";
         AIT_LogMessage(stMsgText);
      endif;
   endif;
}
```

continue ステートメントを if 構造体から削除しました。

AITCE-0041

'< **ループ構造体** >' **が** 255 break **ステートメントを超えました。**

要因

255 より多くの break ステートメントがループ構造体内にあります。ループ構造体（do-while, while，for-next または switch 構造体）内で使用できる break ステートメントの最大数は 255 です。

対処

ループ構造体内で指定する break ステートメントの数を，最大数以下にしてください。

AITCE-0042

'< **ループ構造体** >' **が** 255 continue **ステートメントを超えました。**

要因

255 より多くの continue ステートメントがループ構造体内にあります。ループ構造体（do-while, while，for-next）内で使用できる continue ステートメントの最大数は 255 です。

対処

ループ構造体内で指定する continue ステートメントの数を，最大数以下にしてください。

AITCE-0043

switch **ステートメントの** case **ラベル数が** 255 **を超えています。**

要因

255 より多くの case ラベルが switch ステートメント内にあります。switch ステートメント内で使用できる case ラベルの最大数は 255 です。

対処

switch ステートメント内で指定する case ラベルの数は，最大数以下にしてください。

AITCE-0044

switch **ステートメントに一つ以上の** default **ラベルは使用できません。**

要因

switch ステートメント内で使用できる default ステートメントは一つだけです。

対処

switch ステートメント内で指定する default ステートメントは一つだけにしてください。

例

誤った指定例

```
DEFINE
{
   const string FileVersion = "7.1";
   string stMsgText;
}
MAIN
{
   switch(FileVersion )
      case "7.1":
         ...
         ...
         break;
      default:      // 最初のdefaultです
         ...
         ...
         break;
      default:      // defaultが二つ指定されています
         ...
         ...
         break;
   endswitch;
}
```

各 switch ステートメントで指定できる default ステートメントは一つだけです。この例では，switch ステートメントに default ステートメントが二つ指定されています。

正しい指定例

```
DEFINE
{
   const string FileVersion = "7.1";
   string stMsgText;
}
MAIN
{
   switch(FileVersion)
      case "7.1":
         ...
         ...
         break;
      default:      // defaultを一つだけ指定しました
         ...
         ...
         break;
   endswitch;
}
```

switch ステートメントから default ステートメントを一つ削除しました。

AITCE-0045

パッケージ情報に必要なフィールド '< フィールド名 >' が見つかりません。

要因

PACKAGE_INFO セクションで指定する必要のあるフィールドのうち，どれかが指定されていません。

対処

PACKAGE_INFO セクションで必要なすべてのフィールドを指定してください。

例

誤った指定例

```
PACKAGE_INFO
{
   // 指定する必要のあるPackageIDが欠けています
   Product = "Adobe Acrobat Reader 5.05";
```

```
    Version = "505";
    InstallerName = "Ar505jpn.exe";
    InstallDrive = "C:";
    InstallDirectory = "'¥Program Files'¥Adobe";
}
```

指定する必要のある PackageID が指定されていません。

正しい指定例

```
PACKAGE_INFO
{
    PackageID = "ADOBEACROBATREADER";    // 指定する必要のあるパッケージ項目を指
定しました
    Product = "Adobe Acrobat Reader 5.05";
    Version = "505";
    InstallerName = "Ar505jpn.exe";
    InstallDrive = "C:";
    InstallDirectory = "'¥Program Files'¥Adobe";
}
```

PackageID を PACKAGE_INFO セクション内に追加しました。

AITCE-0046

演算子 '+','-' と '!' は，文字列定数で使用できません。

要因

単項演算子と文字列定数が一緒に指定されています。

対処

単項演算子と文字列定数は一緒に指定しないでください。

例

誤った指定例

```
DEFINE
{
    const integer OK_END =0;
    const integer NG_END = -1;
    const string szMsgText = !"30";  // 「!」を文字列定数に使用しました
}
```

DEFINE セクションで文字列定数を初期化する際，単項演算子（「+」，「-」，および「!」）は使用できません。
ここでは，変数「szMsgText」で，「!」が文字列定数と一緒に指定されています。

正しい指定例

```
DEFINE
{
    const integer OK_END =0;
    const integer NG_END = -1;
    const string szMsgText = "30";      // 「!」を削除しました
}
```

「!」を変数「szMsgText」から削除しました。

AITCE-0047

式内でラベルの使用方法が誤っています。

要因

ラベルが式内で使用されています。

対処

ラベルは式内で使用しないでください。

例

誤った指定例

```
DEFINE
{
   integer sloop_max = 0;
}
MAIN
{
ErrorLabel:
   if (ErrorLabel)      // if構造体内でラベル名が使用されています
      AIT_LogMessage("Setup(Japanese)For Windows-Start");
      if (AIT_FileExists("#setup.exe") == 0)
         goto ErrorLabel;
         sloop_max = 0;
      endif;
   endif;
}
```

ラベルは式内で使用できません。この例では，ラベル「ErrorLabel」が if 構造体内で指定されています。

正しい指定例

```
DEFINE
{
   integer sloop_max = 0;
   string stMsgText;
}
MAIN
{
   if(1)      // ラベルを式から削除しました
      if (AIT_FileExists("#setup.exe") == 0)
         goto ErrorLabel;
         sloop_max = 0;
      endif;
ErrorLabel:
      stMsgText = "Setup(Japanese) " + InstallerName + " Not Found";
      AIT_LogMessage(stMsgText);
   endif;
}
```

ラベル「ErrorLabel」を if 構造体から削除し，式内の値を有効にしました。

AITCE-0048

case 内で ""!"" 演算子は使用できません。

要因

case ラベル値に式が指定されています。switch ステートメント内の case ラベル値には定数値しか指定できません。

対処

case ラベル値内では式を指定しないでください。

例

誤った指定例

```
DEFINE
{
   const integer FileVersion = 7;
   string stMsgText;
}
MAIN
{
   switch (FileVersion)
      case 7:
         ...
         ...
         break;
      case 5 + 1:      // caseステートメント内では式を使用できません
         ...
```

```
            ...
            break;
         default:
            ...
            ...
            break;
      endswitch;
   }
```

switch ステートメント内では定数だけが指定できます。この例では，2 番目の case ラベルに加算演算式が指定されています。

正しい指定例

```
   DEFINE
   {
      const integer FileVersion = 7;
      string stMsgText;
   }
   MAIN
   {
      switch (FileVersion)
         case 7:
            ...
            ...
            break;
         case 6:       // 式を削除し，定数を指定しました
            ...
            ...
            break;
         default:
            ...
            ...
            break;
      endswitch;
   }
```

2 番目の case ステートメントから加算演算式を削除し，定数を指定しました。

AITCE-0050

AIT ファイルのバージョンと実行エンジンの DLL のバージョンが不一致です。

要因

AIT ファイルのバージョンが実行エンジンの DLL より新しいことが考えられます。

対処

AIT ファイルの実行時に，スクリプトエンジンの DLL のバージョンが PACKAGE_INFO セクションで示されている ScriptFileVersion よりも新しいか確認してください。ScriptFileVersion は，DLL のバージョンよりも前である必要があります。

AITCW-0016

< ランクの高いデータ型 > から < ランクの低いデータ型 > に変換しました。データが失われている可能性があります。

要因

AIT ファイルで，ランクの低いデータ型にランクの高いデータ型を割り当てようとすると，値が切り捨てられ，それがランクの低いデータ型に割り当てられます。float 型変数が integer 型変数に割り当てられた場合や，integer 型変数が変数に割り当てられたことが考えられます。

対処

ランクの高いデータ型を使用してください。

例

誤った指定例

```
DEFINE
{
   const integer OK_END = 0;
   const integer SLEEP_TIME = 3.8;      // float型変数を整数定数に割り当てようとしています。値は切り捨てられ，3だけが変数に保存されます。
}
```

ここでは，float 値が integer 型変数「SLEEP_TIME」に割り当てられています。割り当てられた値は切り捨てられ，integer 値だけが変数「SLEEP_TIME」に保存されます。

正しい指定例

```
DEFINE
{
   const integer OK_END =0;
   const float SLEEP_TIME = 3.8;       // 変数の型をfloat型に変更しました
}
```

float 値を保持するために，変数「SLEEP_TIME」のデータ型を integer 型から float 型に変更しました。

AITCW-0017

< 処理 > で bool 型変数は指定できません。

要因

bool 型変数を予期しない方法で指定すると，この警告が呼び出されます。例えば，bool 型変数を除算演算や剰余演算（/，%）に指定すると，0 による除算または 0/0 矛盾につながり，警告メッセージが呼び出されます。

対処

このような場合は，bool 型変数を指定しないでください。

例

誤った指定例

```
DEFINE
{
   const integer Sloop_Max = 30;
   integer sloop_count;
   bool IsPathSet = false;
}
MAIN
{
   sloop_count = sloop_max / IsPathSet;      // bool型変数が除数として指定されています
}
```

bool 型変数を使用すると，0 による除算となり，この警告メッセージが表示されます。ここでは，bool 型変数「IsPathSet」が除数として指定されているため，0 による除算となります。

正しい指定例

```
DEFINE
{
   integer NG_END = 1;
   const integer sloop_max = 30;
   integer sloop_count;
}
MAIN
{
   sloop_count = sloop_max / NG_END;      // bool型変数を削除し，integer型変数を除数として指定しました
}
```

bool 型変数「IsPathSet」を削除し，integer 型変数「NG_END」を除数として指定しました。

AITCW-0028

演算子 < 演算子名 > で < データ型 1> と < データ型 2> は混在できません。

要因

ビットの論理積や論理和などの演算の式で，integer 型や bool 型が混合して指定されています。

対処

式のオペランドでは一貫したデータ型を指定してください。

例

誤った指定例

```
DEFINE
{
   integer sloop_max = 0;
   bool IsPathSet;
}
MAIN
{
  sloop_max = sloop_max & IsPathSet;      // ビット演算の論理積に対してinteger
型とbool型が指定されています
}
```

この例では，「&」演算に対して integer 型と bool 型が指定され，データ型が混在しています。

正しい指定例

```
DEFINE
{
   integer sloop_max = 0;
   integer sloop_count;
}
MAIN
{
   sloop_max = sloop_max & sloop_count;       // ビット演算の論理積に対して
integer型とinteger型を指定しました
}
```

「&」演算に対して同じデータ型（integer 型と integer 型）を指定しました。

AITCW-0033

switch **ステートメントには** default **ラベルしかありません。**

要因

switch ステートメントで default ステートメントの case だけが指定されています。switch ステートメントの case ラベルが指定されていません。これは，ステートメントのシーケンスに相当します。

対処

switch ステートメントで少なくとも一つの case ラベルを指定してください。

例

誤った指定例

```
DEFINE
{
   const string FileVersion = "7.1";
   string stMsgText;
}
MAIN
{
   switch(FileVersion)
      default:      // switchステートメントにdefaultステートメントしか使用してい
ません
         ...
         ...
         break;
   endswitch;
```

```
}
```

すべての switch ステートメントに，一つ以上の case ステートメントが必要ですが，この例では，case ステートメントがなく default ステートメントしか指定されていません。

正しい指定例

```
DEFINE
{
   const string FileVersion = "7.1";
   string stMsgText;
}
MAIN
{
   switch(Version )
     case "7.1":     // switchステートメントにcaseステートメントが含まれています
         ...
         ...
         break;
      default:
         ...
         ...
         break;
   endswitch;
}
```

switch ステートメントに case ステートメントを追加しました。

AITCW-0035

< ラベル名 > : 定義されたラベルが使われていません。

要因

ラベルが定義されていますが，参照されていません。このラベルを無視します。

対処

ラベル名を参照する goto ステートメントを指定してください。

例

誤った指定例

```
DEFINE
{
   const string ExeVersion = "7.1";
   const string FileVersion = "7.1";
   integer sloop_max = 0;
}
MAIN
{
   if (ExeVersion == FileVersion )
ErrorLabel:   // ラベルに関連するgotoステートメントがありません
      AIT_LogMessage("Setup(Japanese)For Windows-Start");
      if (AIT_FileExists("#setup.exe") == 0)
         sloop_max = 0;
      endif;
   endif;
}
```

ラベルには，対応する goto ステートメントが必要です。goto ステートメントなしでラベルステートメントを定義すると，そのラベルステートメントは無視されます。この例では，ラベル「ErrorLabel」が goto ステートメントなしで指定されています。

正しい指定例

```
DEFINE
{
   const string ExeVersion = "7.1";
   const string FileVersion = "7.1";
   integer sloop_max = 0;
}
```

```
MAIN
{
   if (ExeVersion == FileVersion)
      AIT_LogMessage("Setup(Japanese)For Windows-Start");
      if (AIT_FileExists("#setup.exe") == 0)
         goto ErrorLabel;      // gotoステートメントを指定しました
         sloop_max = 0;
      endif;
ErrorLabel:      // ラベルが定義されています
      stMsgText = "Setup(Japanese) " + InstallerName + " Not Found";
      AIT_LogMessage(stMsgText);
   endif;
}
```

ラベル「ErrorLabel」に対応する goto ステートメントを指定しました。

なお，必要のないラベルステートメントを削除する対処方法もあります。

AITCW-0036

< 変数名 > : 定義された変数が使われていません。

要因

変数が DEFINE セクションで定義されていますが，参照されていません。

対処

不要な場合や使用しない場合は，変数の定義を削除してください。

例

誤った指定例

```
DEFINE
{
   const string ExeVersion = "7.1";
   const string FileVersion = "7.1";
   integer sloop_max;      // この変数はプログラム内のどこからも参照されていません
   string stMsgText;
}
MAIN
{
   if (ExeVersion == FileVersion)
      AIT_LogMessage("Setup(Japanese)For Windows-Start");
      if (AIT_FileExists("#setup.exe") == 0)
         stMsgText = "Setup(Japanese) " + InstallerName + " Not Found";
      endif;
   endif;
}
```

上の例で，変数「sloop_max」は DEFINE セクションで定義されていますが，MAIN セクション内で使用されていません。

正しい指定例

```
DEFINE
{
   const string ExeVersion = "7.1";
   const string FileVersion = "7.1";
   integer sloop_max;
   string stMsgText;
}
MAIN
{
   if (ExeVersion == FileVersion)
      AIT_LogMessage("Setup(Japanese)For Windows-Start");
      if (AIT_FileExists("#setup.exe") == 0)
         sloop_max = 0;// ここで変数「sloop_max」が使用されています
         stMsgText = "Setup(Japanese) " + InstallerName + " Not Found";
         AIT_LogMessage(stMsgText);
      endif;
   endif;
}
```

変数「sloop_max」は MAIN セクション内で使用しました。
なお，MAIN セクションで必要のない変数は DEFINE セクションから削除する対処方法もあります。

付録

付録 A　メニュー一覧

［Automatic Installation Tool］ウィンドウのメニュー一覧を次の表に示します。

表 A-1　［Automatic Installation Tool］ウィンドウのメニュー一覧

メニュー		機能	ショートカットキー
ファイル	新規作成	AIT ファイルを新規に作成します。	［Ctrl］+［N］
	開く	既存の AIT ファイルを開きます。	［Ctrl］+［O］
	閉じる	アクティブな AIT ファイルを閉じます。	－
	上書き保存	アクティブな AIT ファイルを上書き保存します。	［Ctrl］+［S］
	名前を付けて保存	アクティブな AIT ファイルに新しいファイル名を付けて保存します。	－
	すべて保存	ウィンドウ上の，変更が加えられたすべての AIT ファイルを保存します。	－
	印刷	アクティブな AIT ファイルを印刷します。	［Ctrl］+［P］
	印刷プレビュー	アクティブな AIT ファイルの印刷イメージを表示します。	－
	印刷設定	［プリンタの設定］ダイアログボックスを表示します。プリンタの設定を変更できます。	－
	最近使ったファイル	最近使ったファイルをリストから選択して表示します。	－
	終了	Automatic Installation Tool を終了します。	［Alt］+［F4］
編集	元に戻す	直前の編集操作を取り消します。	［Ctrl］+［Z］
	やり直し	取り消した直前の編集操作をやり直します。	［Ctrl］+［Y］
	切り取り	文字列を切り取ってクリップボードに貼り付けます。	［Ctrl］+［X］
	コピー	文字列をコピーしてクリップボードに貼り付けます。	［Ctrl］+［C］
	貼り付け	クリップボードの文字列を貼り付けます。	［Ctrl］+［V］
	削除	選択した文字列を削除します。	［Delete］
	すべて選択	AIT ファイル全体を選択します。	［Ctrl］+［A］
	検索	［検索］ダイアログボックスを表示します。指定した文字列を検索します。	［Ctrl］+［F］
	次を検索	カーソル位置から，後方に対して文字列の検索を続行します。	［F3］
	前を検索	カーソル位置から，前方に対して文字列の検索を続行します。	［Shift］+［F3］
	置換	［置換］ダイアログボックスを表示します。検索文字列を指定した文字列に置換します。検索文字列には正規表現を指定できます。	［Ctrl］+［H］
	ジャンプ	［ジャンプ］ダイアログボックスを表示します。指定した行にカーソルを移動します。	［Ctrl］+［G］
表示	ツールバー	次に示すツールバーの表示 / 非表示を切り替えます。 • ［標準］ツールバー • ［ビルド］ツールバー • ［ユティリティ］ツールバー また，［カスタマイズ］ダイアログボックスを表示し，ツールバーおよびコマンドの配置を設定できます。	－
	ステータスバー	ステータスバーの表示 / 非表示を切り替えます。	－

メニュー		機能	ショートカットキー
	アウトプット	アウトプットウィンドウを表示します。	［Alt］+［F2］
	監視ウィンドウ	デバッグ中に，監視ウィンドウを表示します。	［Alt］+［F3］
	ワークブック	• **ワークブックモード** アクティブファイルビュー時のワークブックモードの表示 / 非表示を切り替えます。 • **トグルアイコン** ワークブックモードビューでのファイルのワークブックタブアイコンの表示 / 非表示を切り替えます。	―
ビルド	文法チェック	アクティブな AIT ファイルの文法をチェックし，アウトプットウィンドウに警告とエラーを表示します。	［Ctrl］+［F7］
	実行	アクティブな AIT ファイルの文法をチェックしたあと，実行を開始します。	［Ctrl］+［F5］
デバッグ	実行	現在のステートメントからブレークポイントまで，AIT ファイルを実行します。	［F5］
	デバッグの中断	デバッグを終了し，通常の編集に戻ります。	［Shift］+［F5］
	ステップオーバー	現在のステートメントから次のステートメントまで AIT ファイルを実行します。	［F10］
	カーソル行の前まで実行	カーソルのある行の前まで，AIT ファイルを実行します。カーソル行に一時的なブレークポイントを設定した場合と同じです。	［Ctrl］+［F10］
	ブレークポイントの追加／削除	指定した位置でブレークポイントを追加，削除できます。	［F9］
	ブレークポイントの設定	［ブレークポイントの設定］ダイアログボックスを表示し，ブレークポイントを追加，削除できます。	［Alt］+［F9］または［Ctrl］+［B］
ツール	レコーダ	［レコーダ］ダイアログボックスを表示します。ユーザ操作をレコーディングし，AIT ファイルを自動生成します。	［Ctrl］+［R］
	ウィンドウプロパティ	［ウィンドウプロパティ］ダイアログボックスを表示します。指定したウィンドウまたはコントロールの属性を取得します。	［Ctrl］+［W］
	パッケージ情報	［パッケージ情報］ダイアログボックスを表示します。PACKAGE_INFO セクションを生成します。	［Ctrl］+［P］
	オプション	［オプション］ダイアログボックスを表示します。AIT ファイルの書式，文法チェック時のメッセージ表示数，レコーダのログ出力世代数などを設定できます。	［Ctrl］+［O］
	カスタマイズ	［カスタマイズ］ダイアログボックスを表示します。ツールバーおよびコマンドの配置を設定できます。	［Ctrl］+［C］
ウィンドウ	新規ウィンドウ	新しいアクティブな AIT ファイルを開きます。	［Ctrl］+［N］
	重ねて表示	開かれているすべてのウィンドウを重ねて表示します。	［Ctrl］+［C］
	並べて表示	開かれているすべてのウィンドウを横に並べて表示します。	［Ctrl］+［T］
	アイコンの整列	ウィンドウのアイコンを下段に整列します。	［Ctrl］+［A］
	閉じる	アクティブな AIT ファイルを閉じます。	［Ctrl］+［O］
	すべて閉じる	AIT ファイルをすべて閉じます。	［Ctrl］+［L］
ヘルプ	目次	Automatic Installation Tool ガイドの目次を表示します。	［Ctrl］+［C］
	バージョン情報	Automatic Installation Tool のバージョン情報を表示します。	［Ctrl］+［A］

（凡例）　－：該当なし

付録 B　JP1/NETM/DM で提供する AIT ファイル

JP1/NETM/DM で提供する AIT ファイルの一覧を次の表に示します。

表 B-1　JP1/NETM/DM で提供する AIT ファイル一覧

項番	パッケージ名	バージョン	パッケージ識別 ID	パッケージ識別用ファイル名
1	Windows 2000 Service Pack 3	3	WINDOWS2000SERVICEPACK3	W2Ksp3_jap.exe
2	Windows 2000 Service Pack 4	4	WINDOWS2000SERVICEPACK4	W2KSP4_ja.EXE
3	Windows XP Service Pack 1	1	WINDOWSXPSERVICEPACK1	xpsp1_ja_x86.exe
4	Microsoft Office 2000 Premium ※	2000	MSOFFICE-2000-PREMIUM	APPLAUSE.WAV
5	Internet Explorer 6.0 SP1	6	INTERNETEXPLORER6	Ie6setup.exe
6	McAfee VirusScan 9.0	90	MCAFEE-VIRUSSCAN	ediskimg.cab
7	Norton AntiVirus 2003 ※	2003	NORTON-ANTIVIRUS	NAV_2003.PDF
8	Norton AntiVirus 2004 ※	2004	NORTON-ANTIVIRUS	NAVU2004.PDF
9	Norton AntiVirus 2005 ※	2005	NORTON-ANTIVIRUS	NAV2005.PDF
10	Adobe Acrobat Reader 5.05	505	ADOBEACROBATREADER	Ar505jpn.exe
11	Adobe Reader 6.01	601	ADOBEREADER	AdbeRdr60_jpn_full.exe
12	Microsoft Office XP Professional with FrontPage ※	2002	MSOFFICE-XP-PROFESSIONAL	OFREAD10.HTM
13	Windows Installer 2.0 for Windows NT 4.0 and 2000	2	WINDOWSINSTALLER20-NT	InstMsiW.exe
14	Windows Installer 2.0 for Windows 98 and Me	2	WINDOWSINSTALLER20-98	InstMsiA.exe
15	Windows Script 5.6 for Windows 98 Me and NT 4.0	56	WINDOWSSCRIPT56	scr56jp.exe
16	AIT-HIBUN-REG	0100	AIT-HIBUN-REG	HIBUNREG.BIN
17	AIT-HIBUN-INSTCHK	0100	AIT-HIBUN-INSTCHK	HIBUNCHK.BIN
				HIBUNCHK¥Setup.exe
				HIBUNCHK¥Setup.ssc
18	AIT-HIBUN-INST	0100	AIT-HIBUN-INST	HIBUNINS.BIN
				HIBUNINS¥Setup.exe
				HIBUNINS¥Setup.ssc
19	AIT-HIBUN-LOG	0100	AIT-HIBUN-LOG	HIBUNLOG.BIN

注※　配布先のクライアントに Windows Installer のコンポーネントが必要です。

付録 B.1　JP1/NETM/DM で提供する AIT ファイルの設定内容

JP1/NETM/DM で提供する AIT ファイルを使ったインストール時の，再起動の有無とオプションの設定

内容を次に示します。

（1） Windows 2000 Service Pack 3

- 再起動：なし
- ［ライセンス契約］ダイアログボックス：「同意します」を選択
- ［オプションの選択］ダイアログボックス：「ファイルをアーカイブする」を選択
- ［再起動］ダイアログボックス：「今再起動しない」を選択

（2） Windows 2000 Service Pack 4

- 再起動：なし
- ［使用許諾契約］ダイアログボックス：「同意します」を選択
- ［オプションの選択］ダイアログボックス：「ファイルをアーカイブする」を選択
- ［再起動］ダイアログボックス：「今再起動しない」を選択

（3） Windows XP Service Pack 1

- 再起動：なし
- ［ライセンス契約］ダイアログボックス：「同意します」を選択
- ［オプションの選択］ダイアログボックス：「ファイルをアーカイブする」を選択
- ［再起動］ダイアログボックス：「今再起動しない」を選択

（4） Microsoft Office 2000 Premium

- 再起動：なし
- ［Office メンテナンスモード］ダイアログボックス：「Office の修復」を選択
- ［使用許諾とサポート情報］ダイアログボックス：「「使用許諾契約書」の条件に同意します」を選択
- ［インストールの準備］ダイアログボックス：「今すぐインストール」を選択

（5） Internet Explorer 6.0 SP1

- 再起動：あり
- ［Internet Explorer とインターネットツールの開始］ダイアログボックス：「同意する」を選択
- ［Windows Update: Internet Explorer とインターネットツール］ダイアログボックス：「最小構成インストール，またブラウザのカスタマイズ」を選択
- ［コンポーネントのオプション］ダイアログボックス：「Internet Explorer をインストールするためのフォルダ」にパッケージ情報のインストール先を指定。また，「標準構成」を選択

（6） McAfee VirusScan 9.0

- 再起動：なし
- ［使用許諾契約書］ダイアログボックス：「同意する」を選択
- ［VirusScan インストール ウィザード］ダイアログボックス：「VirusScan デスクトップ アイコンを作成」を選択
- ［世界ウィルス地図］ダイアログボックス：「はい，参加します。」を選択

（7） Norton AntiVirus 2003

- 再起動：なし
- ［ライセンス契約書］ダイアログボックス：「ライセンス契約に同意します」を選択
- ［宛先フォルダ］ダイアログボックス：パッケージ情報のインストール先を設定

（8） Norton AntiVirus 2004

- 再起動：なし
- ［ウイルススキャンしますか?］ダイアログボックス：「いいえ」を選択
- ［使用許諾契約］ダイアログボックス：「使用許諾契約に同意します」を選択

- ［インストール先フォルダを選択］ダイアログボックス：パッケージ情報のインストール先を設定

（9） Norton AntiVirus 2005

- 再起動：なし
- ［ライセンス］ダイアログボックス：「使用許諾契約に同意します」を選択。また，［プロダクトキー］にパッケージ情報のシリアルナンバーを設定※
- ［インストール前のスキャン］ダイアログボックス：「スキャンを開始」を選択しない
- ［インストール先フォルダを選択］ダイアログボックス：パッケージ情報のインストール先を設定

注※　パッケージ情報のシリアルナンバーにはプロダクトキーを設定してください。

（10） Adobe Acrobat Reader 5.05

- 再起動：なし
- ［インストール先の選択］ダイアログボックス：パッケージ情報のインストール先を設定

（11） Adobe Reader 6.01

- 再起動：なし
- ［インストール先フォルダの変更］ダイアログボックス：パッケージ情報のインストール先を設定

（12） Microsoft Office XP Professional with FrontPage

- 再起動：なし
- ［ユーザ情報］ダイアログボックス：「プロダクトキー」にパッケージ情報のシリアルナンバーを設定
- ［使用許諾契約書］ダイアログボックス：「「使用許諾契約書」の条項に同意します」を選択
- ［インストールの種類］ダイアログボックス：「今すぐインストール」を選択
- ［メンテナンスモードオプション］ダイアログボックス：「Office の修復」を選択。また，「インストール先」にパッケージ情報のインストール先を設定
- ［Office の再インストールまたは修復］ダイアログボックス：「Office を再インストールします」を選択

（13） Windows Installer 2.0 for Windows NT 4.0 and 2000

再起動：なし

（14） Windows Installer 2.0 for Windows 98 and Me

再起動：なし

（15） Windows Script 5.6 for Windows 98 Me and NT 4.0

再起動：なし

（16） AIT-HIBUN-REG

再起動：なし

（17） AIT-HIBUN-INSTCHK

再起動：なし

（18） AIT-HIBUN-INST

再起動：なし

（19） AIT-HIBUN-LOG

- 再起動：なし

- ［情報収集ツール］ダイアログボックス：「収集」を選択
- ［ツール実行確認］ダイアログボックス：「同意します」を選択
- ［情報収集ツール］ダイアログボックス：「OK」を選択
- ［情報収集ツール］ダイアログボックス：「キャンセル」を選択

付録 B.2　JP1/NETM/DM で提供する AIT ファイルを使用する場合の注意事項

JP1/NETM/DM で提供する AIT ファイルを使用してパッケージングする場合，パッケージャに「パッケージ名」，「バージョン」，「パッケージ識別 ID」，および「世代番号」が自動的に表示されます。

JP1/NETM/DM で AIT ファイルを提供していない製品をパッケージングした場合に，JP1/NETM/DM で提供される AIT ファイルの情報が表示されるときは，次の手順で対処してください。

1. JP1/NETM/DM **のインストール先ディレクトリ** ¥MASTER¥PPDEFAIT.DMP **を別のフォルダにコピーする。**
2. JP1/NETM/DM **のインストール先ディレクトリ** ¥MASTER¥PPDEFAIT.DMP **から，表示された情報の行を削除する。**
3. **再度パッケージングを実行する。**
4. **別のフォルダにコピーした** PPDEFAIT.DMP **を，**JP1/NETM/DM **のインストール先ディレクトリ** ¥MASTER¥PPDEFAIT.DMP **に上書きコピーする。**

付録 C　PP 識別情報ファイルの編集方法

AIT ファイル作成時に生成された PP 識別情報ファイルは，JP1/NETM/DM のインストール先フォルダ ¥DMPRM に，PPDEFAIT.DMP というファイル名で格納されます。生成された PP 識別情報ファイルを編集する場合は，ここで説明する形式に従って設定してください。

PP 識別情報ファイルの形式は，次のとおりです。

```
情報マップ；パッケージ識別 ID; バージョン；パッケージ名 ;AIT ファイルのフルパス；識別用ファイル名 1; 識別用ファイル名 2; . . . ; 識別用ファイル名 N;
```

PP 識別情報ファイルには，1 行に 1 ソフトウェアの定義を 499 バイト以内で記述します。ソフトウェアが複数ある場合は，複数行記述してください。各項目はセミコロン（;）で区切ります。各項目の詳細は次のとおりです。なお，PP 識別情報ファイルで定義した「パッケージ識別 ID」，「バージョン」，および「パッケージ名」は，パッケージング時に［JP1/NETM/DM パッケージング］ダイアログボックスに表示され，変更できません。

- **情報マップ**（必ず記述）
  000001 を記述します。
- **パッケージ識別 ID**（必ず記述）
  配布するソフトウェアのパッケージ識別 ID です。AIT ファイルで指定したパッケージ識別 ID を記述します。
- **バージョン**（必ず記述）
  配布するソフトウェアのバージョンです。AIT ファイルで指定したバージョンを記述します。
- **パッケージ名**（必ず記述）
  配布するソフトウェアのパッケージ名です。AIT ファイルで指定したパッケージ名を記述します。
- **AIT ファイルのフルパス**（必ず記述）
  AIT ファイルの完全なパス名を，ドライブ名も含めて記述します。256 バイト以内で指定してください。全角文字は使用できません。
- **識別用ファイル名 1 ～ *N***（必ず記述）
  配布するソフトウェアをユニークに識別できるファイル名を記述します。パッケージング時に，このファイルが存在すると，配布するソフトウェアであると判断されます。複数のファイルを記述した場合は，すべてのファイルが存在するとき，配布するソフトウェアであると判断されます。

AIT ファイル「AR505.ais」が「C:¥ADOBE」にある場合の，PP 識別情報ファイルの例を次に示します。

```
000001;ADOBEACROBATREADER;505;Adobe Acrobat Reader 5.05;C:¥ADOBE¥AR505.ais;Ar505jpn.exe;
```

付録 D　レコーダファイルを使ったリモートインストール

レコーダファイルとは，JP1/NETM/DM を使用してソフトウェアを配布する際に，ソフトウェアのインストーラに自動応答するスクリプトのことです。したがって，レコーダファイルを使うとクライアントユーザの手を煩わせることなくソフトウェアをインストールできます。

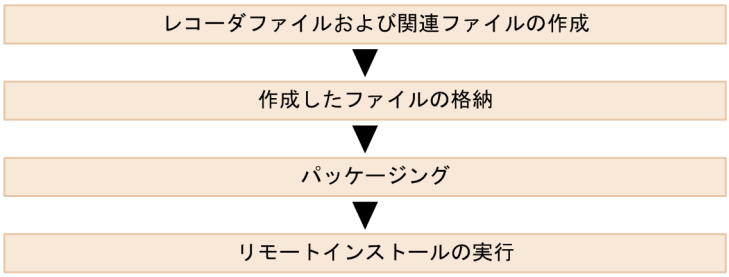
レコーダファイルを標準で用意していないソフトウェアや，GUI でインストールするユーザ作成のプログラムをリモートインストールする場合に，レコーダファイルおよび関連するファイルを作成してインストールする方法を次に示します。

付録 D.1　レコーダファイルを使ったリモートインストールの概要

(1) レコーダファイルを使ったリモートインストールの流れ

レコーダファイルを使ったリモートインストールの流れを次の図に示します。

図 D-1　レコーダファイルを使ったリモートインストールの流れ



(2) レコーダファイルおよび関連ファイルの作成

リモートインストールするソフトウェアのインストール手順を調査してレコーダファイルを作成します。また，関連ファイルも必要に応じて作成してください。

レコーダファイルの関連ファイルには，レコーダファイル本体のほか，「インストール定義ファイル」，「PP 識別情報ファイル」があります。関連ファイルの役割を次に示します。

インストール定義ファイル

他社ソフトウェアをリモートインストールする場合に必要な詳細情報（所有者名，インストール先ディレクトリなど）を定義します。その定義内容は，パッケージング時に表示されるので，ユーザが変更することもできます。

PP 識別情報ファイル

JP1/NETM/DM に対応する他社ソフトウェアであることを示すためのファイルです。パッケージャが，パッケージング時にソフトウェアを自動的に認識するために使用します。

各ファイルの作成方法については，「付録 D.4　関連ファイルの作成手順」を参照してください。

(3) 作成したファイルの格納

レコーダファイルを使ったパッケージのパッケージングには，作成したファイルを所定の場所に格納します。

インストール定義ファイル（INSTABL.DEF）と PP 識別ファイル（PPDEF.DMP）の格納場所は固定です。レコーダファイル（.PCD または .PC6）については任意の場所に格納できます。

各ファイルの格納先については，「付録 D.2(4)　レコーダファイルをパッケージャに組み込む」を参照してください。

(4) パッケージング

レコーダファイルとその関連ファイルを格納後，配布するソフトウェアをパッケージャからパッケージングします。パッケージング時には，「パッケージ名」などのパッケージ情報や「インストールタイミング」などのインストール時のオプションを指定できます。なお，「インストール定義ファイル」で定義した内容は，デフォルトで表示されます。

パッケージングの方法の詳細については，マニュアル「運用ガイド 1」の「2.1　パッケージングの方法」を参照してください。

(a) レコーダファイルの監視時間の設定

レコーダファイルを使用したリモートインストール処理では，インストール中にユーザがダイアログボックスに応答したり，キーボードやマウスを操作したりすると，その時点で手動インストールに切り替わってしまうため，インストール自体が中断してしまうことがあります。このような場合に備えて，インストール中にユーザの応答待ち状態になってから一定の時間が経過したあと，インストールを強制的に中断する時間（レコーダファイルの監視時間）を設定しておくことができます。

レコーダファイルの監視時間の設定方法については，マニュアル「運用ガイド 1」の「2.2.13 ［レコーダファイル設定］パネル」を参照してください。

通常は，レコーダファイルで配布ソフトウェアをインストールした時間の約 3 倍の時間を目安にして設定してください。

なお，インストールが途中で中断するような不測の事態に備えてレコーダファイル自体の構造を工夫しておくこともできます。レコーダファイルの作成方法については，「付録 D.2(2)　レコーダファイルの構造」を参照してください。

(5) リモートインストールの実行

リモートインストールマネージャで配布のジョブを作成して実行してください。リモートインストールの操作については，マニュアル「運用ガイド 1」の「2.3　リモートインストールの実行」を参照してください。

次に，レコーダファイルおよび関連ファイルの作成方法を説明します。レコーダファイルの作成には Visual Test を使用します。

付録 D.2　レコーダファイルの作成手順

レコーダファイルは，基本的に次の手順で作成します。

1. **インストール手順を調査する。**
   リモートインストールするソフトウェアのインストーラを起動して，インストール手順を調査します。
2. **Visual Test でレコーダファイルを作成する。**
   調査したインストール手順を基に，Visual Test の Test Basic を使ってレコーダファイルを作成，コンパイルして，JP1/NETM/DM で使用できるレコーダファイルにします。
3. **パッケージャにレコーダファイルを組み込む。**
   完成したレコーダファイルをパッケージャに組み込みます。

(1) インストール手順を調査する

リモートインストールするソフトウェアのインストーラを起動して，そのインストール手順を調査します。このとき，インストーラが正常に動作しないソフトウェアは，JP1/NETM/DM でのリモートインストールの対象にはなりません。

なお，リモートインストールするソフトウェアが Windows Installer に対応している場合は，ソフトウェアのインストール画面をカスタマイズすることをお勧めします。終了時とエラー発生時にだけダイアログが表示されるようカスタマイズしてください。これによって，調査が必要なインストール手順が少なくなり，レコーダファイルの作成が容易になります。

次の項目について，インストール方法を OS ごとに区別して調べ，インストール手順を紙に記録してください。

- インストール画面の順序と属性
- 各ダイアログボックスの属性

(a) インストール画面の順序と属性

インストーラが，ユーザに対して「何を，どのような順序で，どういった操作を要求してくるか」を調査する必要があります。何回か手動でインストールしてみて調査し，次の項目について一覧表にまとめてください。

● テキスト
Visual Test の「ウィンドウ情報ユーティリティ」を使って調査した「テキスト」を記述します。

● クラス名
Visual Test の「ウィンドウ情報ユーティリティ」を使って調査した「クラス名」を記述します。

● 操作
このダイアログボックスに対する操作（[OK] ボタンをクリックするなど）を記述します。

● 親子属性
このダイアログボックスが，親ダイアログボックスなのか子ダイアログボックスなのかを記述します。

● 親ウィンドウ
このダイアログボックスが子ダイアログボックスの場合，親ダイアログボックスを記述します。

● 備考
特記事項（コメントなど）がある場合に記述します。

なお，インストール方法や，インストールする PC の状態（OS の種類，ハードディスクの空き容量，メモリの空き容量，インストール済みのソフトなど）によって，インストールの操作は変わります。そのため，入念に調査する必要があります。

例えば，Acrobat Reader 5.05 をインストールする場合，次の表に示す一覧になります。

表 D-1　Acrobat Reader 5.05 のインストールの流れ

#	テキスト	クラス名	操作	親子属性	親ウィンドウ	備考
1	Acrobat Reader 5.0.5 のセットアップ	#32770	• [次へ] ボタンをクリックする。	親	なし	なし
2	インストール先の選択	#32770	• 1 回目は [参照] ボタンをクリックする。 • 2 回目は [次へ] ボタンをクリックする。	親	なし	2 回目は #4 のあとへ

#	テキスト	クラス名	操作	親子属性	親ウィンドウ	備考
3	ディレクトリの選択	#32770	• パス名にインストールパスを指定する。	子	#2	インストール先を変更する場合だけ表示
4	セットアップ	#32770	• ［OK］ボタンをクリックする。	子	#3	インストール先を変更する場合だけ表示
5	セットアップの完了	#32770	• ［いいえ，後でコンピュータを再起動します］を選択する。 • ［完了］ボタンをクリックする。	親	なし	なし

(b) 各ダイアログボックスの属性の調査方法

レコーダファイルは，Visual Test の Test Basic 言語で記述します。ボタンをクリックする，文字を入力する，キーを押すなどの操作は，すべて Visual Test で用意されている関数を使います。これらの関数には，引数が必要です。ダイアログボックスのウィンドウハンドル，ウィンドウクラス，タイトルのテキストといったダイアログボックスの属性を指定しなければなりません。そのため，「(a)　インストール画面の順序と属性」で説明した作業のダイアログボックスの属性を調査する場合，Visual Test に付属する「ウィンドウ情報ユーティリティ」を使って記録することをお勧めします。

「ウィンドウ情報ユーティリティ」は，次のように使用します。

1. Visual Test **のプログラムグループから「ウィンドウ情報」を起動する。**
   ［ウィンドウ情報］ダイアログボックスが表示されます。
2. **［ウィンドウ情報］ダイアログボックスの左上にあるアイコンを，属性を調べたいダイアログボックスにドラッグ & ドロップする。**
   次のような情報を取得できます。

図 D-2　［ウィンドウ情報］ダイアログボックスでウィンドウの情報を取得する流れ

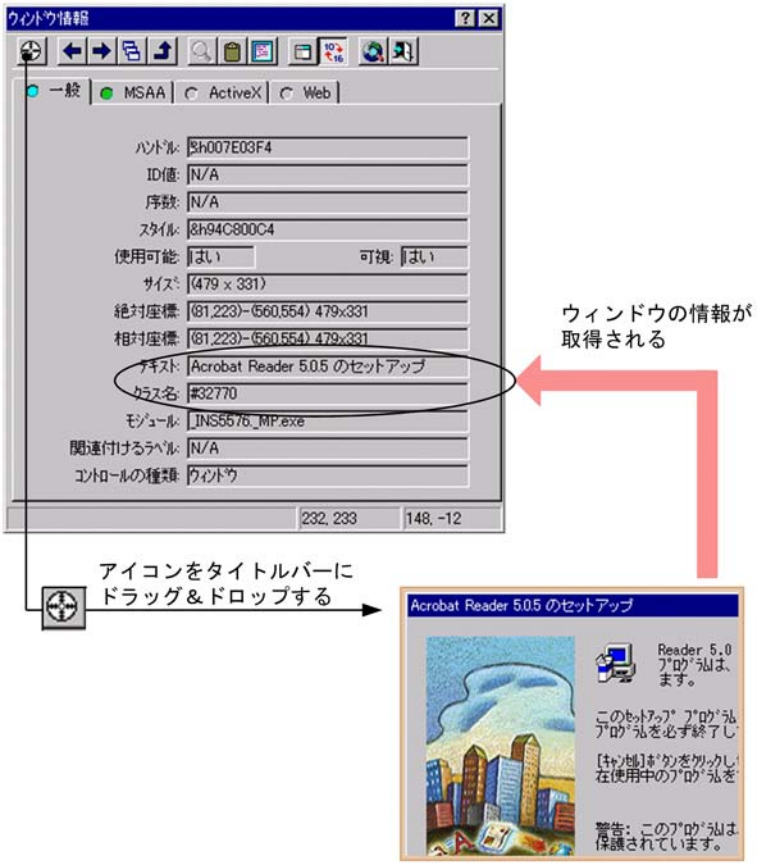
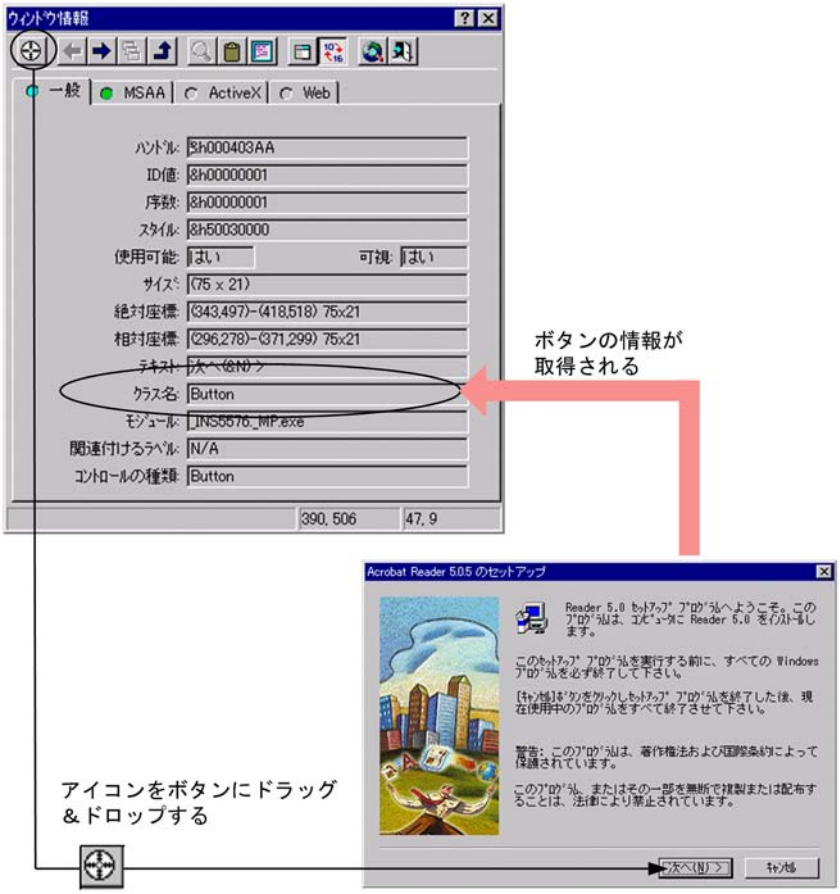


図 D-3 ［ウィンドウ情報］ダイアログボックスでボタンの情報を取得する流れ



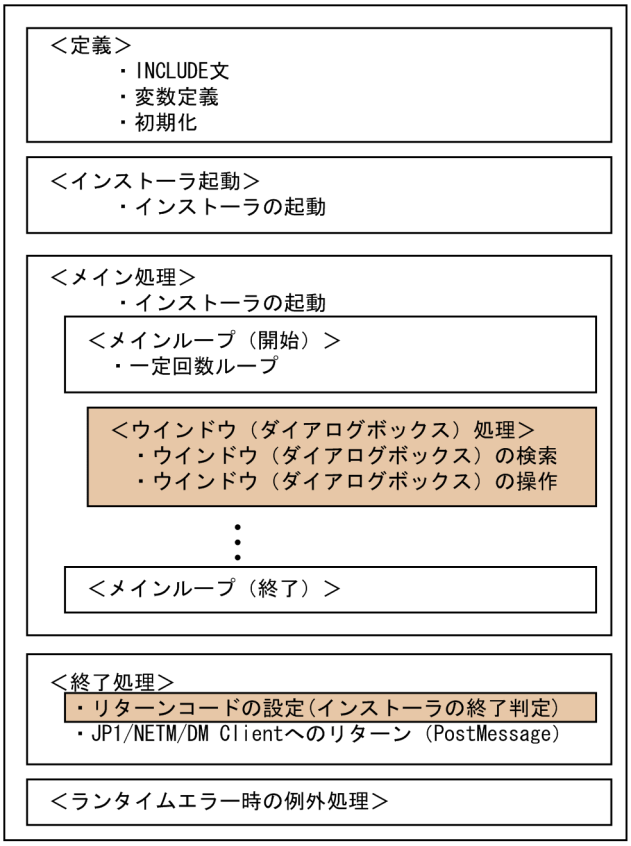
［ウィンドウ情報］ダイアログボックスの［クリップボード］ボタンをクリックすると，情報をクリップボードにコピーできます。テキストエディタなどに貼り付けて，情報をファイルに保存しておくことをお勧めします。

なお，エディットボックスやコンボボックスに対して操作する場合，対象のボックスを示す文字列が必要になります。この文字列を確認する場合にも，「ウィンドウ情報ユーティリティ」を使います。［ウィンドウ情報］ダイアログボックス内の「関連付けるラベル」内の文字列が相当します。

(2) レコーダファイルの構造

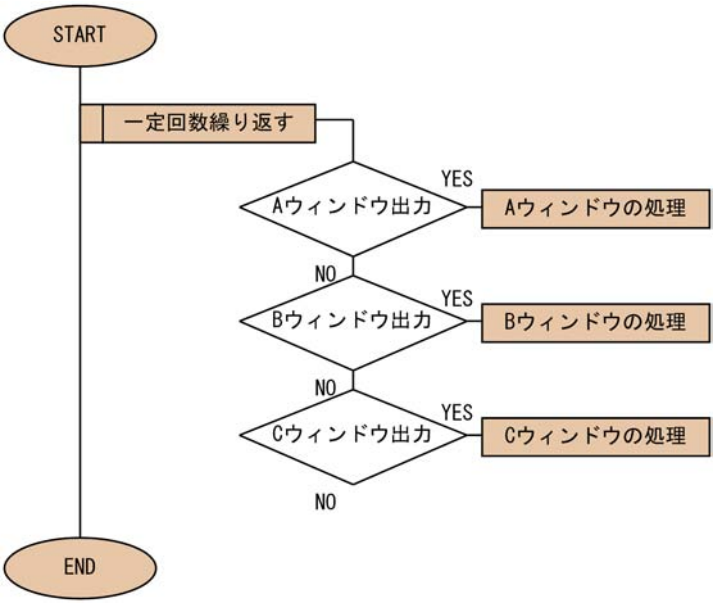
レコーダファイルは次のような構成にするとわかりやすく作成できます。通常は網掛け部分だけをインストーラに合わせて作成してください。

図 D-4　レコーダファイルの構成例



メイン処理の部分は，ユーザが誤って表示されたダイアログボックスに応答したり，キーボードやマウスを操作したりしたため，インストールが中断するような事態に備えて，レコーダファイルを作成することをお勧めします。レコーダファイルの処理例を次に示します。

図 D-5　レコーダファイルの処理例



レコーダファイルは，ループ中で繰り返しダイアログボックスを検索します。したがって，ユーザが誤っ

てAウィンドウに応答してBウィンドウが表示された状態でも，レコーダファイルはAウィンドウの処理をスキップしてBウィンドウの処理を実行できます。

レコーダファイルを作成する場合は出力されるウィンドウを調査し，そのウィンドウに対する処理を並べます。このような処理構造によって出力されるウィンドウの順序やユーザの操作に関係なく処理を完結できます。

(3) Visual Test でレコーダファイルを作成する

記録したインストール手順を基に，Visual Test の Test Basic 言語でレコーダファイルをコーディングします。まず，Visual Test の「シナリオレコーダー」を使ってインストール手順を自動的に生成させます。この自動生成されたコードは，レコーダファイルの原形になります。その後，自動生成されたインストール手順に修正を加え，コンパイルして，JP1/NETM/DM で使用できるレコーダファイルにします。

(a) Visual Test でイベントを記録する

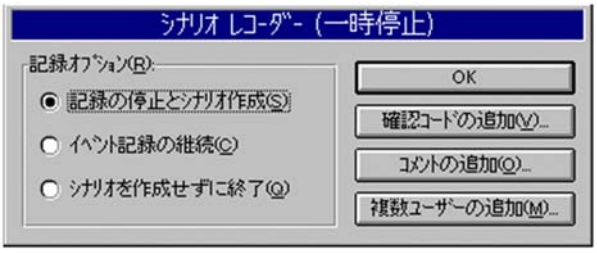
Visual Test の「シナリオレコーダー」を使って，次のような手順でレコーダファイルの原形を作成します。

1. **Visual Test を起動し，メニューで［ツール］から［イベントの記録］を選択する。**
   ［シナリオレコーダー］ダイアログボックスが表示されます。シナリオ名を入力してください。

図 D-6　［シナリオレコーダー］ダイアログボックス

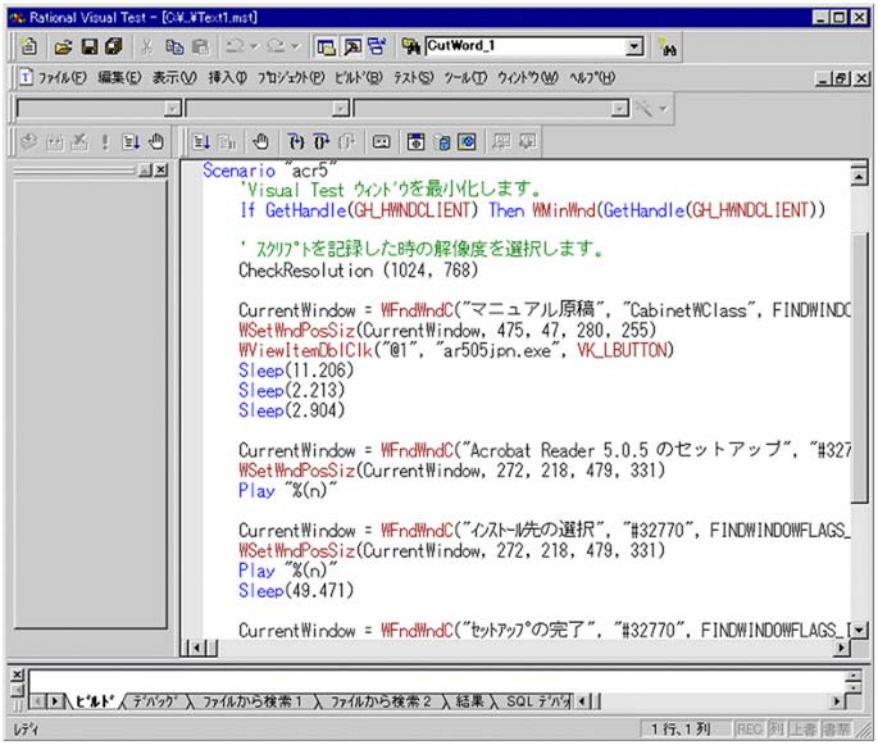
2. **パッケージングしたいソフトウェアを実際にインストールし，イベントを記録する。**
   このインストール作業では，マウスをできるだけ使用しないでください。マウスからの操作は，画面座標に依存してしまうため，レコーダファイルからインストーラへの確実な応答ができなくなってしまいます。マウス操作ではなく，キーボード入力のイベントを記録するようにしてください。
3. **イベントの記録を終了させ，インストール手順のコードを生成する。**
   パッケージングしたいソフトウェアのインストール作業が終わったら，Visual Test のレコーダのアイコンをクリックして，イベントの記録を終了させます。レコーダのアイコンをクリックすると，［シナリオレコーダー（一時停止）］ダイアログボックスが表示されます。

図 D-7　［シナリオレコーダー（一時停止）］ダイアログボックス



［OK］ボタンをクリックすると，自動生成されたイベントが表示されます。

図 D-8　自動生成されたイベントの表示



(b) レコーダファイルの作成手順

レコーダファイルの作成は，インストーラが出力するすべてのダイアログボックスの処理を組み合わせて作成します。

ダイアログボックスの処理は，次の処理を一つのパーツとしてループ中にシーケンシャルに並べます。

- ダイアログボックスの検索
- ダイアログボックスの操作

ダイアログボックスの検索は，WFndWindC 関数を使用します。WFndWindC 関数はダイアログボックスの「テキスト」と「クラス名」を入力情報としています。したがって，出力するダイアログボックスの「テキスト」と「クラス名」を調査する必要があります。

ダイアログボックスを調査するツールには Visual Test の「ウィンドウ情報ユーティリティ」があります。

ダイアログボックスに対する操作には，次のような種類があります。

- アクティブウィンドウ中にボタンやチェックボックスなどの項目があるかないかをチェックする。

- 項目にフォーカスを当てる。
- 項目の状態を確認する。
- ［Enter］キーを押すなどの具体的な操作をする。

シナリオレコーダーで記録したイベントには，ダイアログボックスの情報やそのダイアログボックスに対する処理が記録されています。シナリオレコーダーで記録したイベントの例を次に示します。

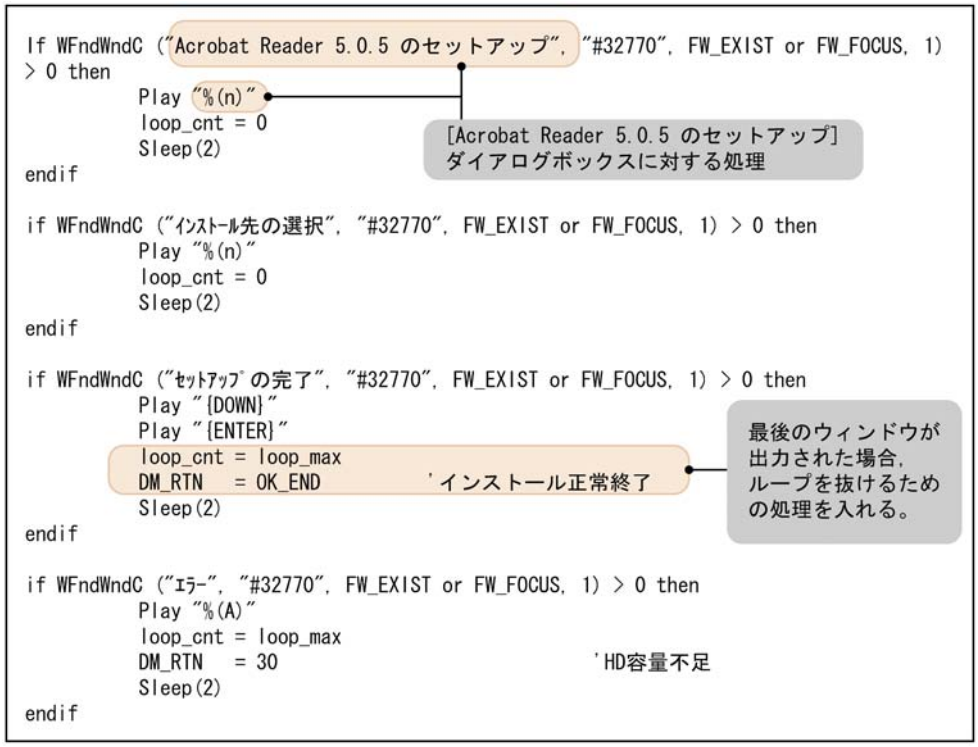
図 D-9　シナリオレコーダーで記録したイベントの例

```
'$INCLUDE 'RECORDER.INC'

SetDefaultWaitTimeout(Timeout)

Scenario "acr5"
          'Visual Test ｳｨﾝﾄﾞｳを最小化します。
          If GetHandle(GH_HWNDCLIENT) Then WMinWnd(GetHandle(GH_HWNDCLIENT))

          ' ｽｸﾘﾌﾟﾄを記録した時の解像度を選択します。
          CheckResolution (1024, 768)

          CurrentWindow = WFndWndC("Data", "CabinetWClass", FINDWINDOWFLAGS_IF, Timeout)
          WSetWndPosSiz(CurrentWindow, 475, 47, 280, 255)
          WViewItemDblClk("@1", "ar505jpn.exe", VK_LBUTTON)
          Sleep(11.206)
          Sleep(2.213)
          Sleep(2.904)

          CurrentWindow = WFndWndC("Acrobat Reader 5.0.5 のセットアップ", "#32770",
FINDWINDOWFLAGS_IF, Timeout)
          WSetWndPosSiz(CurrentWindow, 272, 218, 479, 331)
          Play "%(n)"

          CurrentWindow = WFndWndC("ｲﾝｽﾄｰﾙ先の選択", "#32770", FINDWINDOWFLAGS_IF, Timeout)
          WSetWndPosSiz(CurrentWindow, 272, 218, 479, 331)
          Play "%(n)"
          Sleep(49.471)

          CurrentWindow = WFndWndC("ｾｯﾄｱｯﾌﾟの完了", "#32770", FINDWINDOWFLAGS_IF, Timeout)
          WSetWndPosSiz(CurrentWindow, 272, 218, 479, 331)
          Sleep(5.938)
          Play "{DOWN}"
          Play "{ENTER}"

          CurrentWindow = WFndWndC("Data", "CabinetWClass", FINDWINDOWFLAGS_IF, Timeout)
          WSetWndPosSiz(CurrentWindow, 475, 47, 280, 255)
End Scenario
```

タイトル：Acrobat Reader5.0.5のセットアップ，
クラス：#32770のウィンドウに
対して[N]キーを押す処理

このデータから次のようにレコーダファイルのダイアログボックス処理としてコーディングしてください。

図 D-10　レコーダファイルのダイアログボックス処理としてのコーディング例



レコーダファイルを作成する上でダイアログボックスの検索に必要な WFndWndC 関数の仕様について説明します。

WFndWndC 関数

機能

ウィンドウを検索します。ウィンドウを発見すると，ウィンドウハンドルを取得し，指定によってはそのウィンドウにフォーカスを与えます。

構文：`WFndWndC(text$,classname$,[flags],[timeout&])`

- text$
  検索するウィンドウに結び付けられたテキストを指定します。
- classname$
  検索するウィンドウのクラス名を指定します。ダイアログボックスの場合は「#32770」固定です。
- flags
  FW_FOCUS：ウィンドウを発見した場合にフォーカスを与えます。
  FW_EXIST：検索条件に一致するウィンドウを検索します。ウィンドウを発見した場合，ウィンドウハンドルを返却します。ウィンドウが発見できなかった場合，検索は失敗します。
  FW_ACTIVE_ONLY：アクティブウィンドウだけを検索します。
- timeout&
  ウィンドウの検索に使用できる最大時間を秒単位に指定します。

戻り値

ウィンドウハンドルが long 型または NULL(0) で返ります。

注意

- 子ウィンドウが表示されているため活性化できない状態のダイアログボックスを活性化しようとするとランタイムエラーとなります。

- 連続的に発行すると関数の終了に時間が掛かります。

(c) 複雑なレコーダファイルの作成方法

単純なソフトウェアのインストーラは，単純な構造のレコーダファイルで配布できます。しかし，インストーラによってはもう少し複雑なレコーダファイルを作成する必要があります。

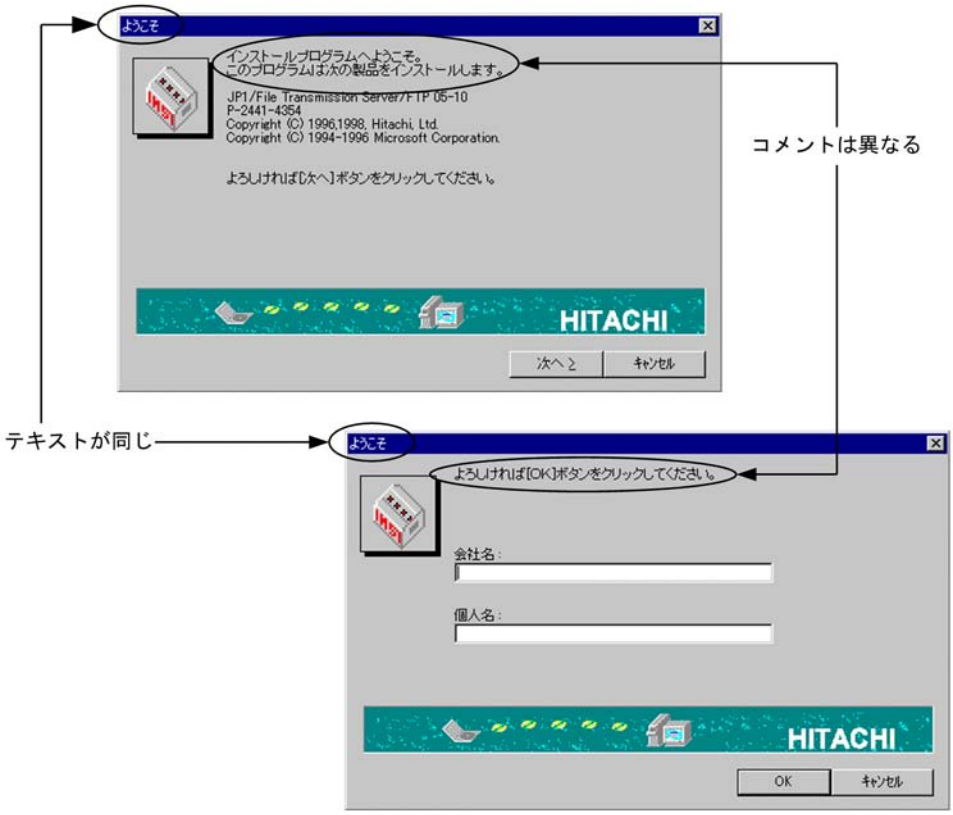
ここでは，同一ウィンドウ名が出力される場合のレコーダファイルの作成方法，および子ウィンドウがある場合のレコーダファイルの作成方法について説明します。

● 同一ウィンドウ名が出力される場合のレコーダファイルの作成方法

レコーダファイルは，あるウィンドウ名に対してどのような応答をするかを決めて作成します。しかし，インストールするソフトウェアのインストーラによっては，同一ウィンドウ名で異なる応答が必要な場合があります。ここでは同一ウィンドウ名の対処方法について説明します。

次のような同一ウィンドウ名で異なる応答が必要な場合は，ウィンドウ内のテキストを判断材料に使います。

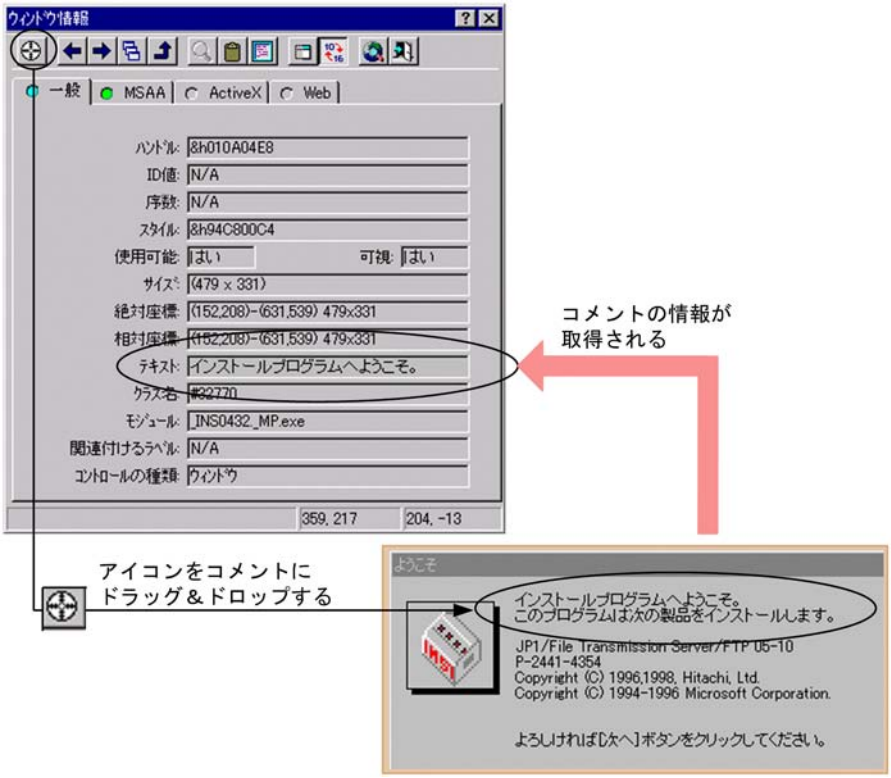
図 D-11　同一ウィンドウ名で異なる応答が必要な画面の例



二つの画面とも「ようこそ」というウィンドウ名ですが，応答は異なります。このような場合は，WStaticExists 関数を使って文字列を検索させます。

コメントの情報は「ウィンドウ情報ユーティリティ」でコメント部分にアイコンをドラッグ＆ドロップすることで取得できます。

図 D-12　［ウィンドウ情報］ダイアログボックスでコメントの情報を取得する流れ



同一ウィンドウ名のインストーラに対するレコーダファイルの作成例を示します。

図 D-13　同一ウィンドウ名のインストーラに対するレコーダファイルの作成例

```
if WFndWndC ("ようこそ", "#32770", FW_EXIST or FW_FOCUS, 1) > 0 then
     '"ようこそ"ウインドウ画面が表示されている場合，画面内のテキストを参照する
   if FLAG1 = PROC_UNEXEC then
      if  WStaticExists ("+インストールプログラムへようこそ") = TRUE  then
        'インストールプログラムへようこそを含む文字列を検索
         WButtonSetFocus("次へ &>")       '[次へ >]ボタンにフォーカスを当てる
         Play "{ENTER}"                   '[次へ >]ボタンを押す
         loop_cnt = 0
         FLAG1 = PROC_EXEC
         Sleep(1)
      endif
   endif
   if FLAG2 = PROC_UNEXEC then
      if WStaticExists ("+よろしければ[OK]ボタンをクリック") = TRUE  then
                        'よろしければ[OK]ボタンをクリックしてください。の文字列を検索
         Play "（株）日立製作所"       '会社名に（株）日立製作所を設定
         Play "{Tab}"
         Play "ソフトウエア事業部"     '個人名にソフトウェア事業部を設定
         WButtonSetFocus("OK")         '[OK]ボタンにフォーカスを当てる
         Play "{ENTER}"                '[OK]ボタンを押す
         loop_cnt = 0
         FLAG1 = PROC_EXEC
         Sleep(1)
      endif
   endif
endif
```

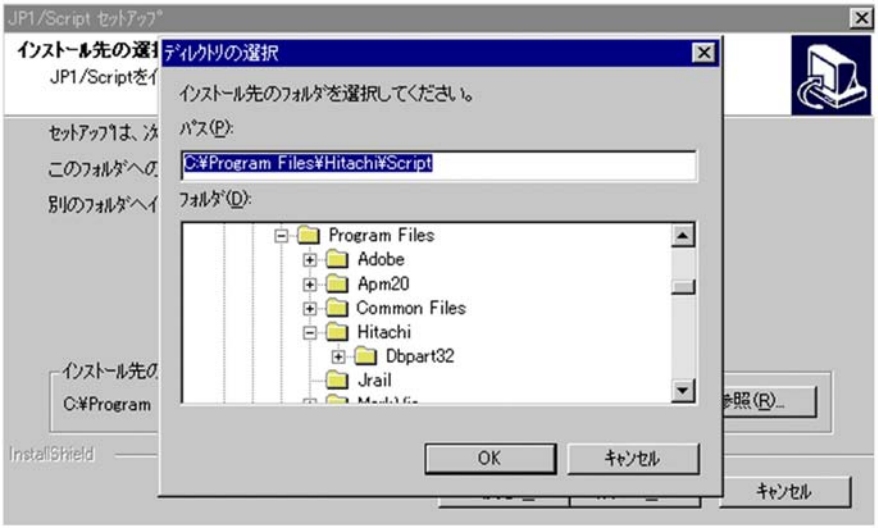
WStaticExists 関数の第 1 引数を「＋」から始めた場合，続きのテキストを含むテキストが存在すると「true」が返されます。長いテキストは「＋」を付けることで前方一致で比較することができます。

このように，同じウィンドウ名の場合は，ウィンドウ内に異なる個所を見つけることで識別させます。そのほかにボタンの名称なども使用できます。

● 子ウィンドウがある場合のレコーダファイルの作成方法

実際のソフトウェアのインストーラにはウィンドウが階層的になっていることがあります。具体的には「子ウィンドウ」と呼ばれ，親ウィンドウからのアクションで表示されます。例えば，次のようなウィンドウのことを示します。

図 D-14　子ウィンドウの例

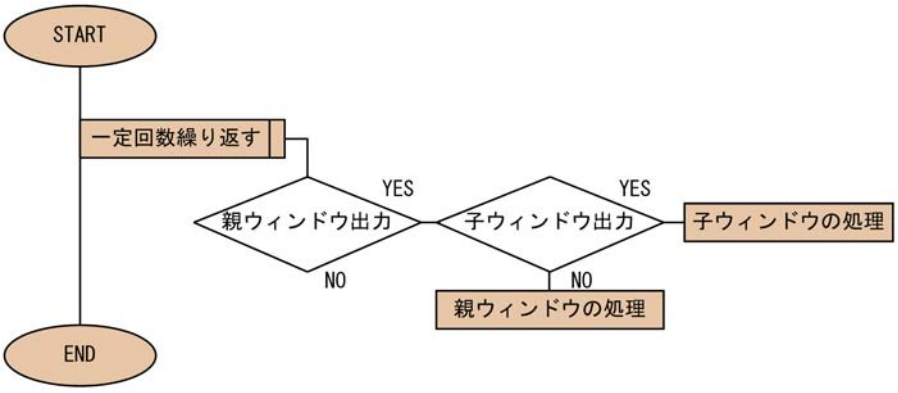


この場合，「JP1/Script セットアップ」が親ウィンドウで「ディレクトリの選択」が子ウィンドウです。

このような環境で親ウィンドウをアクティブにしようとすると，WFndWndC 関数がランタイムエラーとなってしまいます。

子ウィンドウは，親ウィンドウが出力されなければ出力されません。この特性を利用して，親ウィンドウがあるときに子ウィンドウの有無を確認し，子ウィンドウがある場合には子ウィンドウを処理し，子ウィンドウがない場合は親ウィンドウの処理をするようにします。処理例を次に示します。

図 D-15　子ウィンドウがある場合の処理例



具体的なレコーダファイルの記述例を次に示します。

図 D-16　子ウィンドウがある場合のレコーダファイルの記述例

```
whnd=WFndWndC("JP1/Script セットアップ","#32770",FW_EXIST,1) '「親」の確認
if whnd>0 then                                               '「親」があった場合
      work=whnd                                              'ウィンドウハンドルの待避
      whnd=WFndWndC("ﾃﾞｨﾚｸﾄﾘの選択","#32770",FW_EXIST,1)      '「子」の確認
   if whnd>0 then                                            '「子」があった場合
      WSetActWnd(whnd)                                       '「子」をアクティブにする
      play "{ENTER}"                                         ' [ENTER]キーを押す
   else                                                      '「子」がなかった場合
      whnd=work                                              '「親」ハンドルの回復
      WSetActWnd(whnd)                                       '「親」をアクティブにする
      play "%(R)"                                            ' [参照(R)]ボタンを押す
   endif
endif
```

(d) JP1/NETM/DM の拡張機能を使用するための修正

レコーダファイルの中では，Visual Test の多くの関数を使用できます。しかし，これだけでは JP1/NETM/DM と連動させるには不十分なため，JP1/NETM/DM の拡張機能を使用します。そのための修正方法を次に示します。

JP1/NETM/DM の拡張機能を利用するためのファイルをインクルード

レコーダファイルの先頭にある，ほかの $INCLUDE メタコマンドの最後に，JP1/NETM/DM が用意している，ファイルをインクルードするステートメントを追加してください。

```
    :
'$INCLUDE …
'$INCLUDE 'DMPTEST.INC'  ' ここに追加
```

JP1/NETM/DM 固有のグローバル変数を利用するためのサブルーチンを追加

JP1/NETM/DM 固有のグローバル変数を利用するために，DMPSTPRC サブルーチンを呼び出さなければなりません。レコーダファイルでインストーラを起動する前の部分に，DMPSTPRC サブルーチンを追加します。

（例）

```
DMPSTPRC  ' ここに追加
' インストーラを起動します。
run InstallerName, nowait
```

JP1/NETM/DM 固有のグローバル変数の利用

JP1/NETM/DM 固有のグローバル変数を利用して，レコーダファイル内で「インストール先ディレクトリ名」などのインストール情報を参照できます。実際の使用例は，「付録 D.3　レコーダファイルの完成例」を参照してください。使用できるグローバル変数は，次のとおりです。

- InstallerName
  インストール定義ファイルの「InstallerName＝」で指定した，インストールプログラム名。
- InstallDrive
  クライアントの GUI で指定したインストール先ドライブ名。設定値がデフォルト値の場合は，DEFAULT_VALUE になります。
- InstallDirectory
  クライアントの GUI で指定したインストール先ディレクトリ。設定値がデフォルト値の場合は，DEFAULT_VALUE になります。
- InstallPoint
  InstallDrive 変数と InstallDirectory 変数の内容を結合した，インストール位置（ドライブ名を含

むフルパス）。

- InstallUserName
  クライアントの GUI で指定したソフトウェアの所有者名。
- InstallCompanyName
  クライアントの GUI で指定したソフトウェアの所有会社名。

(e) レコーダファイルをコンパイルする

これまでの手順で作成したレコーダファイルのソースをコンパイルします。

1. DMPTEST.INC ファイルをコピーする。
   DMPTEST.INC ファイルをレコーダファイルのあるディレクトリにコピーします。

図 D-17　DMPTEST.INC ファイルのコピー

2. Visual Test のワークベンチからコンパイルする。
   Visual Test のワークベンチから，完成したレコーダファイルのソースをコンパイルしてください。コンパイルが成功すると，コンパイル結果のファイル（*.PC6 ファイルまたは *.PCD ファイル）が生成されます。この生成されたコンパイル結果のファイルが，JP1/NETM/DM で直接使用できるレコーダファイルになります。

(4) レコーダファイルをパッケージャに組み込む

Visual Test でコンパイルしたレコーダファイル（コンパイル結果のファイル）をパッケージャに組み込みます。レコーダファイルを決められたディレクトリに決められた拡張子でコピーすることで，組み込みは終了します。なお，レコーダファイル格納ディレクトリには，レコーダファイル以外のファイルは格納しないでください。

インストール方法が複数あるかどうかで拡張子が変わります。なお，拡張子は，コンパイル時に生成されたデフォルトのままで使用してください。インストール方法が 1 種類の場合の拡張子は，Visual Test 6.0 でコンパイルした場合は「*.PC6」，Visual Test 4.0 でコンパイルした場合は「*.PCD」です。

コピー先のディレクトリを次に示します。説明中の <RecordFileDirectory> という部分は，インストール定義ファイルの RecordFileDirectory 項目で指定するディレクトリです。

インストール方法が一つの場合

<RecordFileDirectory>¥< 任意のファイル名 >.PC6

インストール方法が複数の場合

<RecordFileDirectory>¥< 任意のファイル名 >.1
<RecordFileDirectory>¥< 任意のファイル名 >.2
<RecordFileDirectory>¥< 任意のファイル名 >.3

付録 D.3　レコーダファイルの完成例

Acrobat Reader 5.05 をリモートインストールする場合のレコーダファイルの例を示します。

```
'************************************************************
'レコーダファイル All Rights Reserved, Copyright (C) 2002,Hitachi,Ltd.
```

```
'*****************************************************************
'$INCLUDE 'WINAPI.INC'
'$INCLUDE 'DMPTEST.INC'
Dim WINH      As Long, _
  DM_RTN      As Short, _
  Rtn         As Short, _
  f_DM        As Short, _
  loop_cnt    As Short, _
  loop_max    As Short, _
  whnd      As Long
  const OK_END  =  0 'DM_RTN:正常終了
  SetDefaultWaitTimeout(1)
  ON ERROR GOTO ERROPROC
  '*****************************************************************
  ' 初期化処理
  '*****************************************************************
  DM_RTN   = 10
  f_DM     = 0
  loop_cnt = 0
  loop_max = 30
    DMPSTPRC()
  '*****************************************************************
  ' インストーラ起動処理
  '*****************************************************************
  if 0 = Exists( InstallerName ) then
    loop_max = 0
  end if
  Rtn = Run( InstallerName, SW_NORMAL) ' インストールプログラムの起動
  if Rtn <> 0 then
    loop_max = 0
  endif
  '*****************************************************************
  'メイン処理ループ
  '*****************************************************************
  do while (loop_cnt < loop_max)
    if WFndWndC ("Acrobat Reader 5.0.5 のセットアップ", "#32770", FW_EXIST or
FW_FOCUS, 1) > 0 then
      Play "%(n)"
      loop_cnt = 0
      Sleep(1)
    endif
    if WFndWndC ("セットアップ", "#32770", FW_EXIST or FW_FOCUS, 1) > 0 then
      Play "%(Y)"
      Sleep(1)
    elseif WFndWndC ("ディレクトリの選択", "#32770", FW_EXIST or FW_FOCUS, 1) > 0
then
      Play "%(P)"
      Play "C:¥Program Files¥Adobe¥Acrobat5"
      WButtonSetFocus("OK")
      Play " "
      Sleep(1)
    elseif WFndWndC ("インストール先の選択", "#32770", FW_EXIST or FW_FOCUS, 1) > 0
then
      if f_DM = 9 then
        Play "%(n)"
      elseif f_DM = 0 then
        Play "%(R)"
        f_DM = 9
      endif
      loop_cnt = 0
      Sleep(1)
    endif
      if WFndWndC ("セットアップの完了", "#32770", FW_EXIST or FW_FOCUS, 1) > 0 then
      Play "{DOWN}"
      Play "{ENTER}"
      loop_cnt = loop_max
      DM_RTN   = OK_END  'インストール正常終了
      Sleep(1)
    endif
      if WFndWndC ("エラー", "#32770", FW_EXIST or FW_FOCUS, 1) > 0 then
      Play "%(A)"
      loop_cnt = loop_max
```

```
      DM_RTN   = 30      'HD容量不足
      Sleep(2)
    endif
    loop_cnt = loop_cnt + 1
    Sleep(1)
  loop
  '*****************************************************************
  '終了処理
  '*****************************************************************
  WINH = RegisterWindowMessage("DMP_REC_QUIT")
  if WINH <> 0 then
    PostMessage(HWND_BROADCAST, WINH, DM_RTN, 0)
  endif
End

ERROPROC:
  WINH = RegisterWindowMessage("DMP_REC_QUIT")
  if WINH <> 0 then
    PostMessage (HWND_BROADCAST,WINH,-1,0)
  endif
End
```

付録 D.4　関連ファイルの作成手順

レコーダファイルには,「インストール定義ファイル」と「PP 識別情報ファイル」という二つの関連ファイルがあります。レコーダファイル本体とともに，この二つの関連ファイルも必要に応じて作成してください。

(1) インストール定義ファイルの作成

インストール定義ファイルには，セットアップ方法の種類や FD を入れる順番などを記述します。パッケージャの PC で，次のディレクトリにインストール定義ファイル（INSTABL.DEF）を作成してください。

パッケージャのインストール先ディレクトリ¥DMPRM¥INSTABL.DEF

(a) ファイルの形式

インストール定義ファイルの形式は，次のとおりです。

等号（=）の前後には，1 文字以上の半角スペース，または TAB コードを入力してください。

```
[Package]
PackageID = パッケージ識別 ID
Maker = メーカ名
Version = バージョン
Product = パッケージ名
InstallerName = インストールプログラム名
InstallKind = 番号 . セットアップ方法
InstallDrive = インストールドライブ
InstallDirectory = インストール先ディレクトリ
JUser = 所有者のユーザ名（日本語の場合）
EUser = 所有者のユーザ名（英語の場合）
JCompany = 所有者の会社名（日本語の場合）
ECompany = 所有者の会社名（英語の場合）
SetFdNumber = FD をセットする順番
RecordFileDirectory = レコーダファイルのディレクトリ
RecordFileVersion = Visual Test のバージョン

[Package]
    :
    :
```

インストール定義ファイル内で，各ソフトウェアの定義は [Package] という記述で始めます。ソフトウェアが複数ある場合は，その数だけ [Package] という記述をしてください。

[Package] に続く各項目は，「属性名 = 属性値」という形で定義します。これらの項目をすべて定義する必要はありませんが，必ず定義しなければいけない項目も幾つかあるのでご注意ください。各属性名の詳細を，次に示します。

● PackageID = **パッケージ識別 ID（必ず定義）**

パッケージ識別 ID を指定します。使える文字は，半角の「A ～ Z，-，0 ～ 9」です。指定できる長さは，半角で 1 ～ 44 文字です。

● Maker = **メーカ名**

ソフトウェアのメーカ名を指定します。使える文字に制限はありません。指定できる長さは，半角で 1 ～ 30 文字です。

● Version = **バージョン（必ず定義）**

ソフトウェアのバージョンを指定します。使える文字は，半角の「A ～ Z，0 ～ 9，/」です。指定できる長さは，半角で 1 ～ 6 文字です。なお，世代番号は，「0000」固定になります。

● Product = **パッケージ名（必ず定義）**

パッケージ名を指定します。「¥」は指定できません。指定できる長さは，半角で 1 ～ 50 文字です。

● InstallerName = **インストールプログラム名（必ず定義）**

ソフトウェアをインストールするときの，インストールプログラム（インストーラ）名を指定します。使える文字に制限はありません。指定できる長さは，半角で 1 ～ 256 文字です。媒体が CD-ROM の場合はインストールプログラム名の先頭に，# を付加してください。

（例）

- 媒体が FD の場合

  ```
  InstallerName = INSTALL.EXE
  ```

- 媒体が CD-ROM の場合

  ```
  InstallerName = #INSTALL.EXE
  ```

● InstallKind = **番号 . セットアップ方法**

選択できるセットアップ方法（インストール方法）と，それに対応する番号を次のように指定します。

指定できる長さは，半角で最大 256 文字です。

1. **セットアップ方法** 1, [2. **セットアップ方法** 2, **〜** , *n*. **セットアップ方法** *n*]

n には，1 から順に数字を記述してください。

（例）

```
InstallKind = 1. フルインストール ,2. 最小インストール
```

● InstallDrive = **インストールドライブ（必ず定義）**

ソフトウェアをインストールするドライブ名を指定します。使える文字は，半角の「A 〜 Z，a 〜 z，1 〜 9」です。"1" は，ハードディスクの最初のドライブになります。指定できる長さは，半角で 1 文字です。

● InstallDirectory = **インストール先ディレクトリ（必ず定義）**

ソフトウェアをインストールするディレクトリを ¥ で始まるパス名で指定します。使える文字に制限はありません。指定できる長さは，半角で 1 〜 256 文字です。

● JUser = **所有者のユーザ名（日本語の場合）**

ソフトウェアを所有するユーザの名前を日本語で指定します。使える文字に制限はありません。指定できる長さは，半角で 1 〜 40 文字です。EUser を指定した場合は，指定できません。

● EUser = **所有者のユーザ名（英語の場合）**

ソフトウェアを所有するユーザの名前を英語で指定します。使える文字は，英数字です。指定できる長さは，半角で 1 〜 40 文字です。JUser を指定した場合は，指定できません。

● JCompany = **所有者の会社名（日本語の場合）**

ソフトウェアを所有する会社の名前を日本語で指定します。使える文字に制限はありません。指定できる長さは，半角で 1 〜 80 文字です。ECompany を指定した場合は，指定できません。

● ECompany = **所有者の会社名（英語の場合）**

ソフトウェアを所有する会社の名前を英語で指定します。使える文字は，英数字です。指定できる長さは，半角で 1 〜 40 文字です。JCompany を指定した場合は，指定できません。

● SetFdNumber = FD **をセットする順番（必ず定義）**

FD をセットする順番を次のように指定します。指定できる長さは，半角で最大 256 文字です。媒体が CD-ROM の場合は，SetFdNumber = 0 と指定してください。

SetFdNumber = **番号** [, **番号** , ...]

（例）

1，2，5 枚目の FD をセットする場合は，次のようにします。

```
SetFdNumber = 1,2,5
```

（例）

InstallKind を設定した場合は，次のように SetFdNumber のあとにセットアップ方法の番号を「_番号」の形式で付けてください。

```
SetFdNumber_1 = 1,2,3,4,5
SetFdNumber_2 = 1,2,5
```

● RecordFileDirectory = **レコーダファイルのディレクトリ**

レコーダファイルのあるディレクトリをドライブ名も含めて指定します。使える文字に制限はありません。指定できる長さは，半角で 1 〜 256 文字です。この属性名自体の指定を省略した場合，ジョブ実行時に必ず「クライアントユーザによるインストール」ジョブを指定し，クライアント側でパッケージセットアップマネージャからインストールしてください。「パッケージのインストール」を指定したジョブは最終的にエラーとなります。

● RecordFileVersion = Visual Test **のバージョン（必ず定義）**

レコーダファイルの作成に使用した Visual Test のバージョンを，半角数字 1 けたで次のように設定してください。

- Visual Test 6.0 の場合
  `RecordFileVersion = 6`
- Visual Test 4.0 の場合
  `RecordFileVersion = 4`

(b) インストール定義ファイルの例

インストール定義ファイルの例を次に示します。

```
[Package]
PackageID = ACROBATREADER405
Version = 0405
Product = Acrobat Reader 4.0
InstallerName = #Ar405jpn.exe
SetFdNumber = 0
RecordFileDirectory = c:¥recod¥ar40
RecordFileVersion = 6
```

(2) PP 識別情報ファイルの作成

PP 識別情報ファイルには，パッケージング時に選択するファイルをパッケージに対応付けるための情報を記述します。インストール定義ファイルで定義したパッケージについては，この PP 識別情報ファイルにも必ず定義してください。

PP 識別情報ファイルは，パッケージャの PC で，次のディレクトリに作成してください。

パッケージャのインストール先ディレクトリ `¥DMPRM¥PPDEF.DMP`

(a) ファイルの形式

PP 識別情報ファイルでは，1 行に 1 ソフトウェアの定義を記述します。ソフトウェアが複数ある場合は，複数行記述してください。形式は次のとおりです。

```
情報マップ；パッケージ識別 ID; バージョン番号；パッケージ名；ファイル名；
      :
      :
```

各項目は，セミコロン（;）で区切ります。各項目の詳細は，次のとおりです。

● **情報マップ**
必ず 000001 を指定します。

● **パッケージ識別** ID
インストールするソフトウェアのパッケージ識別 ID を指定します。

● **バージョン番号**
インストールするソフトウェアのバージョンを指定します。

● **パッケージ名**
インストールするソフトウェアのパッケージ名を指定します。

● **ファイル名**
インストールするソフトウェアをユニークに識別できるようなファイル名を指定します。媒体が FD の

場合は，1 枚目の FD にあるファイルの名前を指定してください。通常は，媒体中のソフトウェアの名前を含むようなファイル名になります。

(b) PP 識別情報ファイルの例

PP 識別情報ファイルの例を次に示します。

```
000001;ACROBATREADER405;0405;Acrobat Reader 4.0;Ar405jpn.exe;
```

付録 D.5　レコーダファイル作成後の確認

作成したレコーダファイル，インストール定義ファイル，および PP 識別情報ファイルを使ってソフトウェアをリモートインストールするために，次の項目を最後に確認してください。

(1) レコーダファイルは，正しいディレクトリにあるか

レコーダファイルは，「付録 D.2(4)　レコーダファイルをパッケージャに組み込む」で説明したディレクトリに格納してください。また，ファイル名と拡張子は正しいかどうか確認してください。

(2) インストール定義ファイル，PP 識別情報ファイルは，正しいディレクトリにあるか

インストール定義ファイル，PP 識別情報ファイルは，「付録 D.4　関連ファイルの作成手順」で説明したディレクトリに格納してください。また，正しいファイル名になっているかどうか確認してください。

(3) インストール定義ファイル，PP 識別情報ファイルの内容は正しいか

インストール定義ファイルおよび PP 識別情報ファイルが，「付録 D.4　関連ファイルの作成手順」で説明した構文で，正しく作成されているかどうか確認してください。

(4) レコーダファイルの位置は，インストール定義ファイルで指定されているか

インストール定義ファイルの RecordFileDirectory 項目に，レコーダファイルのあるディレクトリを正しく指定してください。

(5) PP 識別情報ファイルのファイル名が，ほかのソフトウェアと重なっていないか

パッケージングするソフトウェアを正常に認識させるために，重ならないファイル名を PP 識別情報ファイルに付けてください。同じソフトウェアでも，バージョンが異なるときは異なるファイル名を付ける必要があります。

付録 D.6　JP1/NETM/DM で提供するレコーダファイル

JP1/NETM/DM で提供しているレコーダファイルの一覧を次の表に示します。

表 D-2　JP1/NETM/DM で提供するレコーダファイル一覧

項番	パッケージ名	バージョン	パッケージ識別 ID	パッケージ識別ファイル名
1	WindowsNT40ServicePack6	0406	WINNT40SP6	sp6full_i386.exe
2	Adobe Acrobat Reader 4.0	0400	ACROBAT-READER	AR40JPN.EXE
3	Adobe Acrobat Reader 5.0	0500	ACROBAT-READER	ar500jpn.exe
4	インターネット エクスプローラ	05216	MS-IEXPLORE	pncodec.cab
5	WMICORE9598NT	0150	WMICORE	wmicore.exe

項番	パッケージ名	バージョン	パッケージ識別 ID	パッケージ識別ファイル名
6	English Navigator R1.0	0100	ENGLISHNAVIGATOR	_SETUP.001
7	DBPARTNER/Client(DOS/V)DF Drv	0401	P-2263-1744	DBPDFDRV.DL_
8	DBPARTNER ODBC Driver	0103	P-2663-5514	DBP0250.002

各レコーダファイルを使ったインストール時の，再起動の有無とオプションの設定内容を次に示します。

（1） WindowsNT40ServicePack6

- 再起動：あり
- ［Windows NT 4.0 Service Pack 6 セットアップへようこそ］ダイアログボックス：「同意する」を選択。また，「後でこの Service Pack をアンインストールできるように，必要なファイルをバックアップする」は，パッケージ情報の「セットアップ方法」の設定によって次のようになる
  - 「1. アンインストールディレクトリあり」を設定：選択する
  - 「2. アンインストールディレクトリなし」を設定：選択しない

（2） Adobe Acrobat Reader 4.0

- 再起動：なし
- ［インストール先の選択］ダイアログボックス：パッケージ情報のインストール先を設定

（3） Adobe Acrobat Reader 5.0

- 再起動：なし
- ［インストール先の選択］ダイアログボックス：パッケージ情報のインストール先を設定

（4） インターネット エクスプローラ

- 再起動：あり
- ［Internet Explorer とインターネットツールのセットアップへようこそ］ダイアログボックス：「同意する」を選択
- ［Windows Update: Internet Explorer とインターネットツール］ダイアログボックス：「最小構成インストール，またブラウザのカスタマイズ」を選択
- ［コンポーネントのオプション］ダイアログボックス：「Internet Explorer をインストールするためのフォルダ」にパッケージ情報のインストール先を設定。また，「標準構成」を選択

（5） WMICORE9598NT

- 再起動：なし
- ［使用許諾契約］ダイアログボックス：「契約に同意します」を選択

（6） English Navigator R1.0

- 再起動：なし
- ［English Navigator システムの設定］ダイアログボックス：パッケージ情報のインストール先を設定

（7） DBPARTNER/Client(DOS/V)DF Drv

再起動：なし

（8） DBPARTNER ODBC Driver

再起動：なし

付録 E　Windows Installer に対応したソフトウェアのリモートインストール手順

Windows Installer は，Microsoft Windows に搭載された新しいインストールサービスです。Windows Installer に対応したソフトウェアは，インストーラの画面を表示させないでインストールすること（サイレントインストール）ができます。サイレントインストールの場合は，インストール時の設定内容をコマンドのオプションやトランスフォーム（.mst ファイル）を使って指定できます。そのため，AIT ファイルを作成する場合に，インストーラの画面を調査したり，画面ごとの処理を定義したりする必要がありません。

JP1/NETM/DM は，Windows Installer に対応したソフトウェア用の AIT ファイルのテンプレートを用意しています。このテンプレートを必要に応じて変更して使用することで，Windows Installer に対応したソフトウェアのリモートインストールが容易にできます。なお，この AIT ファイルのテンプレートを使ってリモートインストールするには，クライアントの PC に，バージョン 07-10 以降のクライアントと Windows Installer 2.0 以降が必要です。

Windows Installer 2.0 は，Windows Server 2003，Windows XP，および Windows 2000（Service Pack 3 以降）には標準でインストールされています。それ以外の OS の場合は，Microsoft 社のホームページで Windows Installer 2.0 を入手し，インストールしておく必要があります。なお，JP1/NETM/DM は，Windows Installer 2.0 のコンポーネントを配布するための AIT ファイルを提供しています。この AIT ファイルを使うことで，クライアントへの Windows Installer 2.0 コンポーネントの配布が容易になります。Windows Installer 2.0 コンポーネントの配布用 AIT ファイルについては，「付録 B　JP1/NETM/DM で提供する AIT ファイル」を参照してください。

Windows Installer に対応したソフトウェアのリモートインストールは，次の図に示す手順で行います。

図 E-1　Windows Installer に対応したソフトウェアのリモートインストールの流れ

付録 E.1　MSIEXEC コマンドのコマンドラインの調査

Windows Installer 対応のソフトウェアをサイレントインストールする場合は，インストール方法や，ユーザ名，インストール先ディレクトリなどの情報を Windows Installer の MSIEXEC コマンドのオプションに指定します。

JP1/NETM/DM を使ってインストールする場合は，MSIEXEC コマンドを呼び出す処理を AIT ファイルの中に定義します。

そのため，MSIEXEC コマンドにどのようなオプションを指定するかを決めておく必要があります。MSIEXEC コマンドにどのようなオプションが指定できるかについては，Windows のオンラインヘルプを参照してください。

指定できる主なオプションの例を次に示します。なお，AIT ファイルを作成するときは，使用するオプションを Windows のオンラインヘルプで必ず確認してください。

（例 1）ユーザーインターフェースを一切表示しない

```
MSIEXEC.EXE /qn /I Example.msi
```

（例 2）ステータスメッセージ，メモリ不足メッセージ，およびエラーメッセージをログファイル（MSIEXEC.LOG）に出力する

```
MSIEXEC.EXE /I Example.msi /Lime MSIEXEC.LOG
```

（例 3）トランスフォーム（TransformList.mst）を利用する

```
MSIEXEC.EXE /I Example.msi TRANSFORMS=TransformList.mst
```

付録 E.2　JP1/NETM/DM が提供しているテンプレート

JP1/NETM/DM では，Windows Installer に対応したソフトウェアをサイレントインストールするために，AIT ファイルのテンプレートを 2 種類，用意しています。テンプレートは Microsoft Office Project 2003 用と Microsoft Office 2003 Resource Kit 用の 2 種類ですが，テンプレートをカスタマイズすれば，ほかのソフトウェアのインストールにも使えます。この 2 種類のテンプレートは，テンプレート中で定義されている処理が異なります。ほかのソフトウェア用にカスタマイズするときは，実行したい処理に近い方のテンプレートを使用してください。

(1) 各テンプレートの特長

2 種類のテンプレートの仕様を次の表に示します。

表 E-1　JP1/NETM/DM が提供する 2 種類のテンプレート

定義項目	Microsoft Office Project 2003 用 テンプレート	Microsoft Office 2003 Resource Kit 用 テンプレート
パッケージ名	Microsoft Office Project Standard 2003	Microsoft Office 2003 Resource Kit
バージョン	2003	2003
パッケージ識別 ID	MSOFFICE-2003-PROJECT-STANDARD	MSOFFICE-2003-RESOURCE-KIT
インストーラ名	PRJSTDE.MSI	ORK.MSI
テンプレートファイル名	インストール先ディレクトリ ¥MASTER¥DMAIT¥TEMPLATE¥OFFICE-2003¥Office2003PrjStandard.ais	インストール先ディレクトリ ¥MASTER¥DMAIT¥TEMPLATE¥OFFICE-2003ORK¥Office2003RKTOOLS.ais
実行するコマンドライン	MSIEXEC.EXE /qn /i PRJSTDE.MSI /Lime MSIEXEC.LOG TRANSFORMS=TransformList.mst REBOOT=R	MSIEXEC.EXE /qn /i OSK.MSI
テンプレートに定義されている処理	• インストーラ画面を表示しない。 • ログファイルを作成し，ステータスメッセージ，メモリ不足メッセージ，エラーメッセージを出力する。 • トランスフォームを利用する。 • インストール後に再起動しない。	• インストーラ画面を表示しない。 • ログファイルを作成しない。 • トランスフォームを利用しない。 • インストール後の再起動を抑制しない。

Microsoft Office Project 2003 用テンプレートには，オプションやトランスフォームを指定してインストールするための処理が定義されています。設定を細かく定義してインストールしたい場合に向いています。

Microsoft Office 2003 Resource Kit 用テンプレートには，最小限のオプションだけが定義されています。このテンプレートは，あまりカスタマイズすることなく，最小限の設定でインストールしたい場合に向いています。

(2) テンプレートの定義内容

ここでは，Microsoft Office Project 2003 用のテンプレートを例に，テンプレートの定義内容をセクションごとに説明します。

(a) PACKAGE_INFO セクション

インストールするソフトウェアのパッケージ情報とセットアップに必要な情報を指定するセクションです。

別のソフトウェアをインストールする場合は，このセクションの変更が必要です。

Microsoft Office Project 2003 用テンプレートの PACKAGE_INFO セクションを次に示します。左側の数字はテンプレートの先頭からの行数です。

図 E-2　Microsoft Office Project 2003 用テンプレートの PACKAGE_INFO セクション

```
1: PACKAGE_INFO
2: {
3:   PackageID         = "MSOFFICE-2003-PROJECT-STANDARD";
4:   Product           = "Microsoft Office Project Standard 2003";
5:   Version           = "2003";
6:   InstallerName     = "PRJSTDE.MSI";
7:   InstallDrive      = "C:";
8:   InstallDirectory  = "'¥Program Files'¥Microsoft Office";
9: }
```

(b) DEFINE セクション

変数や定数を定義するセクションです。このセクションには，トランスフォームのファイル名，MSIEXEC コマンドに指定するオプション，インストールするソフトウェアのプロパティ名などが定義されています。これらを変更したい場合は，このセクションの変更が必要です。

Microsoft Office Project 2003 用テンプレートの DEFINE セクションを次に示します。左側の数字はテンプレートの先頭からの行数です。

図 E-3　Microsoft Office Project 2003 用テンプレートの DEFINE セクション

```
10: DEFINE
11: {
12:   integer  WINH;
13:   integer  iErrorNo;
14:   string   strRegKeyName;
15:   string   strRegValueName;
16:   string   strRegValueData;
17:   string   strNetmdmPath;
18:   string   strLogLine;
19:   string   strParameter;
20:   string   strExecCmdLine;
21:   string   strErrorTxt;
22:   bool     bRtn;
23:
24:   const string BIN      = "'¥BIN";
25:   const string RUNFILE  = "'¥dmpwinis.exe";
26:   const string LOG      = "'¥LOG";
27:   const string MSIEXEC  = "MSIEXEC.EXE";
28:   const string SPACE    = " ";
29:   const string DQUOT    = "'"";
30:   const string BSDQUOT  = "'¥'"";
31:   const string LOGFILE  = "'¥MSIEXEC.LOG";
32:
33:   ////////////////////////////////////////////////////////////////////////
34:   // 以下の定義を変更して利用してください。
35:   // トランスフォーム名
36:   const string TRANSFORMLIST  = "TransformList.mst";
37:
38:   // コマンドラインオプション
39:   const string Option_qn     = "/qn";    // ユーザインターフェイスは表示されません。
40:   const string Option_i      = "/i";     // パッケージを使用してアプリケーションを
インストールします。
41:   const string Option_Lime   = "/lime";  // ログファイルに記録する情報を指定します。
42:
43:   // コマンドラインプロパティ
44:   const string Prop_reboot   = "REBOOT=R";          // すべての再起動を抑制します。
45:   const string Prop_company  = "COMPANYNAME=";      // 会社名を指定するプロパティ
です。
46:   const string Prop_instdir  = "INSTALLDIR=";       // インストール先を指定するプ
ロパティです。
47:   const string Prop_instloc  = "INSTALLLOCATION="; // インストール先を指定するプ
ロパティです。
48:   const string Prop_pidkey   = "PIDKEY=";           // ボリューム ライセンス キー
を指定するプロパティです。
49:   const string Prop_trans    = "TRANSFORMS=";       // トランスフォームを指定する
プロパティです。
50:   const string Prop_user     = "USERNAME=";         // ユーザー名を指定するプロ
パティです。
51:
52:   ////////////////////////////////////////////////////////////////////////
53: }
```

(c) MAIN セクション

インストール時に行う処理を定義するセクションです。このセクションには，Windows Installer の MSIEXEC コマンドを呼び出す処理などが定義されています。MSIEXEC コマンドのコマンドラインを変更する場合には，このセクションの変更が必要です。

Microsoft Office Project 2003 用テンプレートの MAIN セクションを次に示します。左側の数字はテンプレートの先頭からの行数です。

図 E-4　Microsoft Office Project 2003 **用テンプレートの** MAIN **セクション**

```
 54: MAIN
 55: {
 56:   AIT_InitLog(InstallerName);
 57:   // JP1/NETM/DM固有のグローバル変数を取り込みます。
 58:   AIT_DMPSTRC();
 59:
 60:   // JP1/NETM/DM Clientインストール先ディレクトリを取得します。
 61:   strRegKeyName = "SOFTWARE¥HITACHI¥NETM/DM/P";
 62:   strRegValueName = "PathName";
 63:   strRegValueData = "";

  :    ... (中略)

 80:   ////////////////////////////////////////////////////////////////////////
 81:   // 以下の処理を変更して利用してください。
 82:
 83:   // MSIEXEC.EXEのコマンドラインオプションおよびプロパティを作成します。
 84:   // 例) MSIEXEC.EXE /qn /i [MSIファイル名.msi] /lime [ログファイル名] REBOOT=R
 85:   strParameter = "";
 86:   strParameter = MSIEXEC + SPACE + Option_qn;  // /qオプションの追加
 87:   strParameter = strParameter + SPACE + Option_i + SPACE + BSDQUOT + InstallerName +
BSDQUOT;  // /iオプションの追加
 88:   strParameter = strParameter + SPACE + Option_Lime + SPACE + BSDQUOT + strLogLine +
BSDQUOT;  // /lオプションの追加
 89:   // TRANSFORMSプロパティの追加
 90:   strParameter = strParameter + SPACE + Prop_trans + BSDQUOT + TRANSFORMLIST +
BSDQUOT;
 91:   // REBOOTプロパティの追加
 92:   strParameter = strParameter + SPACE + Prop_reboot;
 93:
 94:   // COMPANYNAMEプロパティの追加
 95:   // strParameter = strParameter + SPACE + Prop_company + BSDQUOT + JCompany +
BSDQUOT;
 96:   // USERNAMEプロパティの追加
 97:   // strParameter = strParameter + SPACE + Prop_user + BSDQUOT + JUser + BSDQUOT;
 98:   // INSTALLLOCATIONプロパティの追加
 99:   // strParameter = strParameter + SPACE + Prop_instloc + BSDQUOT + InstallDrive +
InstallDirectory + BSDQUOT;
100:   // PIDKEYプロパティの追加
101:   // strParameter = strParameter + SPACE + Prop_pidkey + BSDQUOT + SerialNumber +
BSDQUOT;
102:
103:   ////////////////////////////////////////////////////////////////////////
104:
105:
106:
107:   // JP1/NETM/DM Clientツールのコマンドラインオプションを作成します。
108:   // [JP1/NETM/DM Clientインストール先ディレクトリ]¥BIN¥dmpwinis.exe [MSIEXEC.EXEの
コマンドラインオプション]
109:   strExecCmdLine = strNetmdmPath + SPACE + DQUOT + strParameter + DQUOT;
110:
111:   // JP1/NETM/DM Client(dmpwinis.exe)経由でWindowsInstallerを起動します。
112:   bRtn = AIT_Exec(strExecCmdLine, SW_NORMAL);

      ... (以下略)
```

なお，AIT ファイルから MSIEXEC コマンドを呼び出す場合，MSIEXEC コマンドの実行結果を取得するために，DMPWINIS コマンドを介して呼び出します。AIT ファイルを新しく作成する場合も MSIEXEC コマンドのコマンドラインを，DMPWINIS コマンドの引数として指定してください。

このスクリプトの実行結果は，保守コードおよび保守コードのユーザステータスとして返されます。このスクリプトが返す実行結果については，「付録 E.7　リモートインストールの実行」を参照してください。

付録 E.3　テンプレートのカスタマイズ

ここでは，Microsoft Office Project 2003 用のテンプレートを例として，テンプレートをカスタマイズする手順を説明します。なお，カスタマイズするときに変更が必要な個所には，コメントとして「以下の定義（処理）を変更して利用してください」と記述されています。変更個所を探す際の目安にしてください。

(1) プロパティの値を変更する

下記の四つのプロパティの値を変更する手順を次に示します。

- 会社名
- ユーザ名
- インストール先ディレクトリ
- CD キー

テンプレートでは，これらのプロパティを設定する処理は，MAIN セクションにコメントとして書かれています。そのため，プロパティを変更する場合は，該当個所のコメントを表す記号を削除する必要があります。プロパティの値は，パッケージ情報ツールを使って，JP1/NETM/DM のグローバル変数に定義します。

プロパティを変更する場合の手順を次に説明します。

1. **パッケージ情報ツールで，変更したいプロパティの値を入力する。**
   パッケージ情報ツールの使用方法については，「2.5　PACKAGE_INFO セクションを生成する」を参照してください。
2. **テンプレートの** MAIN **セクションで，下記に示す行の先頭にある「//」を削除する。**

```
94:    // COMPANYNAMEプロパティの追加
95:    // strParameter = strParameter + SPACE + Prop_company + SPACE + BSDQUOT + JCompany
+ BSDQUOT;
96:    // USERNAMEプロパティの追加
97:    // strParameter = strParameter + SPACE + Prop_user + SPACE + BSDQUOT + JUser +
BSDQUOT;
98:    // INSTALLLOCATIONプロパティの追加
99:    // strParameter = strParameter + SPACE + Prop_instloc + SPACE + BSDQUOT +
InstallDrive + InstallDirectory + BSDQUOT;
100:   // PIDKEYプロパティの追加
101:   // strParameter = strParameter + SPACE + Prop_pidkey + SPACE + BSDQUOT +
SerialNumber + BSDQUOT;
```

//を削除する

(2) トランスフォームの設定を変更する場合

テンプレートでは，MSI ファイルと同じディレクトリにある，TransformList.mst という名称のファイルをトランスフォームとして利用するように定義されています。トランスフォームの設定を変更したい場合は，次のようにしてください。

● トランスフォームを使用しない場合
MAIN セクションにある下記の行の行頭に「//」を挿入して，コメントにします。

```
89:    // TRANSFORMSプロパティの追加
90:    strParameter = strParameter + SPACE + Prop_trans + SPACE + BSDQUOT + TRANSFORMLIST
+ BSDQUOT;
```

//を挿入する

● トランスフォームのファイル名を変更する場合
トランスフォームのファイル名は，テンプレートの 36 行目で定義されています。使用するトランスフォームのファイル名を，パッケージングするディレクトリからの相対パスで指定してください。

(3) コマンドラインの指定内容を変更する場合

(1), (2) の変更以外に，コマンドラインに指定するオプションやプロパティを追加・変更したい場合は，DEFINE セクションおよび MAIN セクションを変更して，実行したいコマンドラインを定義してください。

(4) ほかのソフトウェアをインストールする場合

テンプレートをカスタマイズしてほかのソフトウェアをインストールするための AIT ファイルを作成する場合は，インストールするソフトウェアのパッケージ情報をテンプレートの PACKAGE_INFO セクションに指定します。PACKAGE_INFO セクションを変更するには，パッケージ情報ツールを使用すると便利です。パッケージ情報ツールの使用方法については，「2.5　PACKAGE_INFO セクションを生成する」を参照してください。

Windows Installer に対応したソフトウェアの場合，「インストールプログラム名」には MSI ファイルの名称を指定します。パッケージングするディレクトリからの相対パスで指定してください。

MSIEXEC コマンドのコマンドラインに指定するオプションやプロパティは，インストールするソフトウェアによって異なります。そのソフトウェアのコマンドラインに合わせて，DEFINE セクションや MAIN セクションに変更を加えてください。また，トランスフォームのファイル名や格納ディレクトリについても，インストールするソフトウェアに合わせて変更してください。

(5) AIT ファイル作成時の注意事項

AIT ファイル中で MSIEXEC コマンドを使う場合は，DMPWINIS コマンドの引数に MSIEXEC コマンドのコマンドラインを指定してください。DMPWINIS コマンドの使用例を次に示します。

図 E-5　DMPWINIS コマンドの使用例

```
DMPWINIS.EXE "MSIEXEC.EXE /qn /I ¥"C:¥Program Files¥Hitachi¥NETMDMP¥WORK¥SAMPLE.MSI¥
" /Lime ¥"C:¥Program Files¥Hitachi¥NETMDMP¥LOG¥MSIEXEC.LOG¥" TRANSFORMS=TRANSFORM¥
SAMPLE.MST"
```

インストールの実行結果（成功／失敗）は DMPWINIS コマンドが取得し，クライアントに結果を通知します。インストールが成功すると 0 が，失敗すると 0 以外の値が通知されます。AIT ファイル中に実行結果を通知する処理を記述する必要はありません。ただし，DMPWINIS コマンドの起動成功／失敗は，AIT ファイルで結果を通知してください。

AIT ファイルで DMPWINIS コマンドを使っている場合，AIT ファイル中の AIT_PostMessage を使って送信する結果通知コードには，16 進数で 01 ～ 29 または D0 ～ FF を使用してください。これ以外の値は，JP1/NETM/DM が予約しています。

付録 E.4　PP 識別情報ファイルの作成

通常の AIT ファイルを作成する場合と同様に，PP 識別情報ファイルを作成してください。JP1/NETM/DM が提供しているテンプレートをそのまま使用する場合も，PP 識別情報ファイルの作成は必要です。PP 識別情報ファイルの作成方法の詳細は，「1.2.1　AIT ファイルおよび PP 識別情報ファイルの作成」を参照してください。

● Microsoft Office Project 2003 用テンプレートの場合（C:¥Program Files¥HITACHI¥NETMDMP にインストール）

```
000001;MSOFFICE-2003-PROJECT-STANDARD;2003;Microsoft Office Project Standard 2003;C:¥Program
Files¥HITACHI¥NETMDMP¥MASTER¥DMAIT¥TEMPLATE¥OFFICE-2003¥Office2003PrjStandard.ais;PRJSTDE.MS
I※;
```

注※　識別用ファイル名は，必要に応じて変更してください。

● Microsoft Office 2003 Resource Kit 用テンプレートの場合（C:¥Program Files¥HITACHI¥NETMDMP にインストール）

```
000001;MSOFFICE-2003-RESOURCE-KIT;2003;Microsoft Office 2003 Resource Kit;C:¥Program
Files¥HITACHI¥NETMDMP¥MASTER¥DMAIT¥TEMPLATE¥OFFICE-2003ORK¥Office2003RKTOOLS.ais;ORK.MSI※;
```

注※　識別用ファイル名は，必要に応じて変更してください。

付録 E.5　作成したファイルの格納

作成した PP 識別情報ファイルと AIT ファイルを格納してください。

PP 識別情報ファイルは，パッケージャの PC の次のディレクトリに格納してください。

パッケージャのインストール先ディレクトリ ¥DMPRM¥PPDEFAIT.DMP

また，ユーザが作成した AIT ファイルは，パッケージャの PC の，PP 識別情報ファイルで定義している場所に格納してください。

付録 E.6　パッケージング

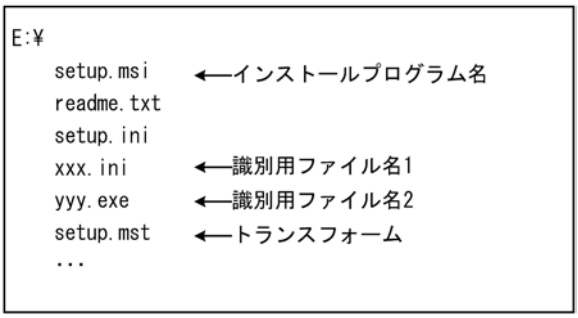
AIT ファイルと PP 識別情報ファイルを所定の場所に格納したあと，配布するソフトウェアをパッケージャでパッケージングします。パッケージングの方法は通常と同じです。

トランスフォームをサーバの共有フォルダなどに格納し，クライアントから参照させる場合は，MSIEXEC コマンドのコマンドラインにトランスフォームの固定パスを指定してください。

トランスフォームを MSI ファイルと一緒に配布する場合は，媒体に格納されているファイルをハードディスクなどにコピーし，トランスフォームと一緒にパッケージングしてください。このとき，トランスフォームは，MSI ファイルから参照できるディレクトリに格納してください。MSIEXEC コマンドのコマンドラインで指定する TRANSFORMS プロパティには，パッケージングするディレクトリからの相対パスを指定してください。

パッケージングするファイルが E ドライブに入っていると仮定して，AIT ファイルおよび PP 識別情報ファイルでの指定方法を説明します。

● トランスフォームを MSI ファイルと同じディレクトリに格納した場合



パッケージングするディレクトリ

```
E:¥
```

AIT ファイルのパッケージ情報で指定する「インストールプログラム名」

```
setup.msi
```

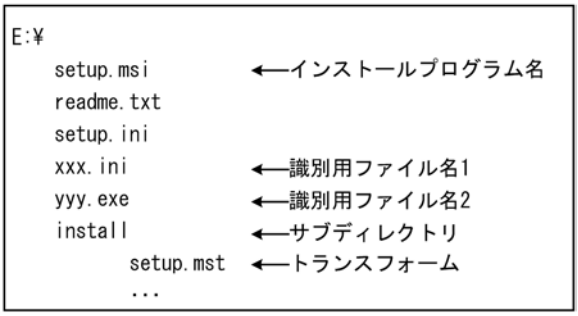
PP 識別情報ファイルで指定する「識別用ファイル名」

```
xxx.ini;yyy.exe
```

MSIEXEC コマンドのコマンドラインに指定する TRANSFORMS プロパティ

```
TRANSFORMS="setup.mst"
```

● トランスフォームを MSI ファイルのサブディレクトリに格納した場合



パッケージングするディレクトリ

```
E:¥
```

AIT ファイルのパッケージ情報で指定する「インストールプログラム名」

```
setup.msi
```

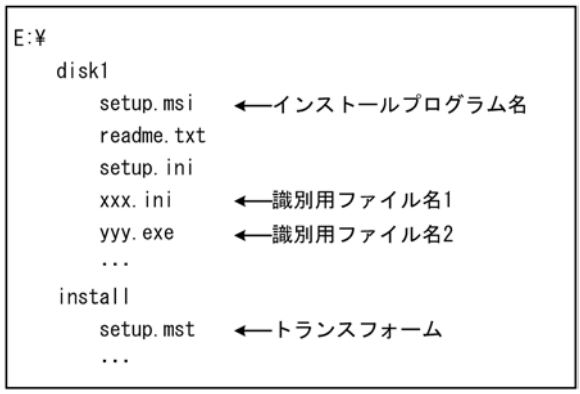
PP 識別情報ファイルで指定する「識別用ファイル名」

```
xxx.ini;yyy.exe
```

MSIEXEC コマンドのコマンドラインに指定する TRANSFORMS プロパティ

```
TRANSFORMS="install¥setup.mst"
```

● トランスフォームを MSI ファイルと同じ階層のディレクトリに格納した場合



パッケージングするディレクトリ

```
E:¥
```

AIT ファイルのパッケージ情報で指定する「インストールプログラム名」

```
disk1¥setup.msi
```

PP 識別情報ファイルで指定する「識別用ファイル名」

```
disk1¥xxx.ini;disk1¥yyy.exe
```

MSIEXEC コマンドのコマンドラインに指定する TRANSFORMS プロパティ

```
TRANSFORMS="install¥setup.mst"
```

付録 E.7　リモートインストールの実行

リモートインストールマネージャを使って，ジョブを作成，実行します。リモートインストールマネージャでの操作については，マニュアル「運用ガイド 1」の「2.3　リモートインストールの実行」を参照してください。

ここで説明したテンプレートを使ってリモートインストールを実行した場合，ジョブの実行結果は，保守コードおよび保守コードのユーザステータスで返されます。

このテンプレートを実行したときに返される保守コードと，それぞれの要因を次の表に示します。

表 E-2　テンプレート実行時の保守コードと要因

保守コード	要因
900000000000	インストール処理が成功しました。 クライアントユーザによるインストールで，ユーザが「インストール成功」を設定した場合を含みます。
30009F07*xx*00 （*xx* はユーザステータス）	インストール処理に失敗しました。 詳細な要因は，ユーザステータスの値でわかります。 • ユーザステータスの値 B0：Windows Installer がインストールされていません。または，DMPWINIS コマンドの引数に誤りがあります。 B1：DMPWINIS コマンドの引数が指定されていません。 C0：DMPWINIS コマンドの起動時にエラーが発生しました。
30009F0B0000	AIT ファイルの処理がタイムアウトしました。
3000AD010000	クライアントユーザによるインストールで，ユーザが「インストール失敗」を設定しました。

付録 F　各バージョンの変更内容

(1) 09-50 の変更内容

● 機能追加なし。

(2) 09-12 の変更内容

● 機能追加なし。

(3) 09-10 の変更内容

● 機能追加なし。

(4) 09-01 の変更内容

● 適用 OS に Windows 7 および Windows Server 2008 R2 を追加した。

● ウィンドウ処理の注意事項を追加した。

● API 関数名「AIT_GetOSType」の Windows Server 2008 の「nMinorVersion（出力用）」の値を「1」から「0」に変更した。

(5) 09-00 の変更内容

● JP1/NETM/DM シリーズのマニュアルで用語を統一した。

(6) 08-52 の変更内容

● 64 ビットバージョン JP1/NETM/DM Client の名称を，64 ビット版 JP1/NETM/DM Client に変更した。

(7) 08-50 の変更内容

● 適用 OS に Windows Server 2008 を追加した。

● 次の API に Windows Server 2008 の説明を追加した。
- AIT_GetOsType
- AIT_TaskbarItemClk
- AIT_TaskbarItemExists
- AIT_TaskbarItemIndex
- AIT_TaskbarItemSelected

(8) 08-11 の変更内容

● 対象製品に，Windows Vista 版の JP1/NETM/DM Client - Operation Log Feature を追加した。

(9) 08-10 の変更内容

● AIT ファイル作成時に，パッケージ情報ツールから PP 識別情報ファイルを生成できるようにした。

● Windows Vista に対応した次のプログラムを対象製品に追加した。
- JP1/NETM/DM Client
- JP1/NETM/DM Client - Base
- JP1/NETM/DM Client - Delivery Feature
- JP1/NETM/DM Client - Remote Control Feature

● AIT ファイル編集時のメッセージ「AITG123-E」,「AITG124-E」,「AITG125-E」を追加した。

(10) 08-02 の変更内容

● AIT 言語で使用する API の説明で，機能説明と形式説明の順序を入れ替えた。

(11) 08-00 の変更内容

● JP1/NETM/DM が提供する AIT ファイルに，秘文連携機能で使用するファイルを追加した。

(12) 07-53 の変更内容

● 適用 OS に Windows Server 2003 (x64) を追加した。

(13) 07-50 の変更内容

● JP1/NETM/DM が提供する AIT ファイルを追加した。

(14) 07-11 の変更内容

● AIT ファイルを作成および使用する場合の注意事項を追加した。

● JP1/NETM/DM が提供する AIT ファイルを追加した。

(15) 07-10 の変更内容

● Windows Installer を使用してインストールするプログラムをサイレントインストールできるようにした。

● Windows Installer のモジュールを配布するための AIT ファイルを提供した。

付録 G　用語解説

（ア行）

イベント

ユーザの操作やプログラムの状態変化などのコンピュータが検知できる事象です。ユーザ操作によって発生するイベントとしては，マウスクリック，メニュー選択，キーボード操作，時間経過などがあります。そのほかのイベントは，未処理例外や，ウィンドウの作成および消滅などのように，アプリケーションの内部またはオペレーティングシステムの内部で発生するものです。

（カ行）

関連付けるラベル

コントロールの一種です。このテキストをユーザが操作したり変更したりすることはできません。一般に，関連づけるラベルは，キャプションが付いていないリストボックスなど，別のコントロールの説明としても使用されます。タブオーダーで，コントロールの直前のテキストが関連づけるラベルになります。

キャプション

コントロールの上部またはウィンドウのタイトルとして表示されるテキストです。

子ウィンドウ

別のウィンドウ（親ウィンドウ）の子孫に当たるウィンドウです。例えば，一般にアプリケーションは，子ウィンドウを使って，一つの親ウィンドウのクライアントエリアを複数の作業エリアに分割します。ダイアログボックスの内部では，チェックボックスやテキストボックスなどの各種コントロールが，ダイアログボックスの子ウィンドウに相当します。

コントロール

テキストボックスやコマンドボタンなど，ウィンドウ上に配置されているグラフィカルオブジェクトのことです。コントロールは，コントロールを表示しているウィンドウの子ウィンドウに相当します。

（タ行）

タブオーダー

タブキーを押して，フォーカスをあるコントロールから次のコントロールへと移動させていく際の順序のことです。

（ナ行）

日時ピッカー

コントロールの一種で，日付または時刻を選択できます。

（ハ行）

ハンドル

ユニークな 4 バイトの整数値です。これを使用してウィンドウやコントロールを識別してアクセスできます。この値は，オペレーティングシステムによって割り当てられています。

フォーカス

ユーザーインターフェースオブジェクト（ウィンドウ，ビュー，ダイアログボックス，ボタンなど）の一時的な属性で，

フォーカスされたオブジェクトは，ユーザ入力を受け取ることができます。例えば，テキストボックスが「フォーカスされた」とき，ユーザが文字列を入力すると，その文字列がテキストボックスに表示されます。通常フォーカスは，強調表示で示されます。

索引

A

B

C

D

E

F

G

I

J

M

P

R

S

V

W

あ

い

う

え

お

か

き

く

け

こ

さ

し

す

せ

そ

た

ち

つ

て

と

に

は

ひ

ふ

へ

ま

む

め

も

ゆ

よ

ら

り

れ

ろ