

IBM Host On-Demand バージョン 13.0

Host On-Demand マクロ・プログラミング・ガイド



IBM Host On-Demand バージョン 13.0

Host On-Demand マクロ・プログラミング・ガイド

本書は、IBM[®] Host On-Demand バージョン 13.0 (IBM Host Access Client Package for Multiplatforms V7.0、プログラム番号 5724-I20)、および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示さ れたりする場合があります。

- 原典: SC31-6378-07 IBM Host On-Demand Version 13.0 Host On-Demand Macro Programming Guide Fifth Edition (February 2016)
- 発行: 日本アイ・ビー・エム株式会社
- 担当: トランスレーション・サービス・センター

Copyright International Business Machiens Corporation 2004, 2016.

© Copyright IBM Corporation 2004, 2016.

目次

本書について.............	. ix
その他の Host On-Demand 資料	. ix
本書の規則	. xi
 第 1 部 マクロの基本	. 1
第 1 章 概要	. 3
Host On-Demand マクロ	. 3
マクロの定義	. 3
マクロの利点	. 3
初級ユーザー	. 3
上級ユーザー	. 3
プログラミング機能...........	. 4
サンプル..............	. 4
マクロの配置	. 4
マクロを使用したエンタープライズ・アプリケーショ	Э
ンの統合	. 4
Host Access Toolkit	. 5
マクロとセキュリティー	. 5
本書における 3270 アプリケーション	. 6
第2章 マクロのコンポーネント	. 7
	7
マクロ・マネージャー	. 7
マクロ・マネージャー・ツールバー	. 7
マクロ・エディター	. 9
コード・エディター	. 10
マクロ・ランタイム	. 11
マクロ・オブジェクト	. 11
その他の用語の定義	. 11
	10
	13
単純マクロの記録	. 13
早純マクロの再生	. 17
キーの組み合わせれのマクロの割り目し	. 18
第4章 マクロの構造	19
マクロ・スクリプト	. 19
XML エレメント	. 19
マクロ・スクリプトの概念視点......	. 20
マクロ・タブの概要	. 22
マクロ画面とそのサブコンポーネント	. 24
アプリケーション画面.........	. 24
マクロ画面	. 25
マクロ画面の概念視点.........	. 26
画面タブの概要	. 27
 第 2 部 マクロの開発	31

第	Į	5	章	Ē	ř–	タ	퐻	Ī,	ì	寅	算	子、	ł	5 6	にて	たべ	<u>`</u>			33
マ	ク	<u>р</u>	形	式の)選	択														33
	基	本	マ	ク	口形	式	と	拡	張	マ	ク	口形	気	のと	比較	Ξ.				33
	ス	• •	IJ	$\boldsymbol{\nu}_{i}$	グと	特	殊	文	字	の	表	記、	演	算	子文	字(のI	取り		
	扨	とい	۰.																	33
	別	の	形	式・	への	7	ク		の	変	換									35
標	準	デ	_ ;	タ型	<u>U</u> .															35
	フ	ř	ル	• •	デー	タ														36
	敷	数	[.																	36
	倍	뛔	度																	36
	ス		IJ	$\boldsymbol{\Sigma}$	グ															36
フ	1		ル	ド																37
値	n	ull	Ι.																	37
算	術	演	算	子ま	らよ	びョ	£													37
	淮	貸	子	お.	よび	式														37
	演	貸	式	の 1	使用	場	所													38
ス	١ ٢	ij	$\frac{1}{2}$	グ運	車結 :	宙貧	≨∃	۲۰	(+)										38
	演	貸	子	お	よび	走			(·	<i>.</i>										38
条	伴	演	、 算二	7. Z	:論	理》	」 宙貨	· 〔1〕	Fι	Б.	۲	ど式								39
	. 条	件	式	に	は複	合	条	伴	を	含	む	こと	が	でき	きる					39
	条	件	式	の	用途															39
自	動	デ	_ /	タ西	一変	愌														40
	J	レン	テ	+	スト	の	影	響												40
	フ	ř	ル	\sim	の変	換														40
	敷	数	(^	の	変換															40
	倍	뛔	度	<u>へ</u>	の変	換														40
	ス	. ト	IJ	$\boldsymbol{\nu}_{i}$	ガヘ	の	変	擙												41
	変	烫換	II	ラ・																41
等	価		•																	41
行	ま	た	はふ	71] ()負(の値	直の	りえ	氢明	未										41
第	(6	章	7	マク		•	Ξ	5:	ン	夕	11	ろん	٦J	にる	$\overline{\mathbf{b}}$	ゥケ	ᄓ		
画	面	īσ.	つ刻	ΔŦ	∎方	法	-													43
概	要																			43
1-74	何	או	L	τ1	使用	す	る	シ	ナ	1)	オ			÷						43
	7	・一	D	可	面の	妧	理	ス	,テ	Ĺ	ジ								•	44
	、ス	テ	·	ジ	1 0	ノご の言	羊紙	Ħ	<i>_</i>			•	•	•	•	•	•	•	•	44
	7	۶'n	ヤ	- ス	全体	сн С	3~	- ・ つ(カ	· 全	ス-	F.—	・ ジ۱	ന	. 一概	要	•	•	•	45
	想	 F更	こ の	結	上 rT 論	(0	-		-		,		-)	.,	1-74		·	·	•	46

フロセス全体 (3 つの全ステージ) の概要 .		. 45
概要の結論・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・		. 46
ステージ 1: 次に処理するマクロ画面を決定する	5.	. 46
有効な次画面のリストへのマクロ画面名の追	加	
(ステップ 1(a))		. 46
画面認識 (ステップ 1(b)).......		. 48
有効な次画面のリストからの候補マクロ画面	jの彳	Ż
前の除去 (ステップ 1(c)).......		. 50
ステージ 2: 選択された候補を新しい現行マクロ	コ面	面
にする	•	. 50
ステージ 3: 新しい現行マクロ画面のアクション	ンを	実
行する		. 50
アクション後の遅延の挿入		. 51

処理サイクルの繰り返し............	51
マクロの終了...............	51
第7章 画面記述と画面認識	53
用語の定義	53
記述タブの概要	54
記述タブの例...............	54
記述の記録	56
記録される情報	56
記録された記述が機能する理由	57
記録されたティスクリフターか提供するフレーム	-0
リーク · · · · · · · · · · · · · · · · · · ·	58
71 ヘクリンス の計画	58
プロヤスの概要	58
個々のディスクリプターの評価	59
デフォルト結合メソッド	59
uselogic 属性................	61
<i>т</i> ́гдо́ IJ Јуде	62
概要	62
「Field Counts and OIA」ディスクリプター. .	63
ストリング・ディスクリプター (<string> エレメ</string>	
ント)	69
カーソル・ディスクリプター (<cursor> エレメン</cursor>	
	73
属性ナイスクリノター ($< attrib> エレメント$)	74
	75
$\lambda \times \lambda \times \lambda$	75
変数更新アクション ($<$ varundate> エレメント)	75
記述内の変数更新アクションの処理	76
uselogic 属性を使用した変数更新	76
	77
(税安	77
(Q配別の) クラヨク	78
アクションのパラメーターの指定	78
「アクション (Actions)」タブの概要	78
「アクション (Actions)」タブの例	78
新規アクションの作成...........	80
アクション	81
ボックス選択アクション (<boxselection> エレメ</boxselection>	
u i i)	81
通信待機アクション (<commwait> エレメント)</commwait>	82
条件アクション (<if> エレメントおよび <else></else></if>	~ (
$ \pm (/////) + (////////////////////////////$	84
抽出アクション (<extract> エレメント) FileUpload アクション (<fileupload> エレメン</fileupload></extract>	87
ト)	93
入力アクション (<input/> エレメント)	97
メッセージ・アクション (<message> エレメン</message>	
h) .	104
マワス・クリック・アクション (<mouseclick></mouseclick>	
	105
$\Sigma \cup \Sigma \cup \Sigma \cup \Sigma$	105

実行アクション (<perform> エレメント)</perform>	107
PlayMacro アクション (<playmacro> エレメン</playmacro>	
h)	109
印刷アクション (<print> エレメント)</print>	111
プロンプト・アクション (<prompt> エレメン</prompt>	
\flat)	114
プログラム実行アクション (<runprogram> エレ</runprogram>	
メント)	119
SQLQuery アクション (<sqlquery> エレメント)</sqlquery>	121
トレース・アクション (<trace> エレメント)</trace>	129
変数更新アクション (<varupdate> エレメント)</varupdate>	131
Xfer アクション (<filexfer> エレメント)</filexfer>	134
第9章 画面認識、パート 2 ・・・	137

7-	3	무		비뱐	비 다	아마	×.		`	1.	~		-	•		•	•	137
有	効な	こ次に	面面	đ														137
入	り口	画	面、	出	口	画	面、	お	よ	びー	→時	画	面					139
	入	り口	画	面														139
	出	コ面	面															140
	\rightarrow	侍画	面															140
画	面彰	認識の	のう	マイ	Ц	7	ウト	、設	定									142
	画	面認	識															142
	画	面間	の	タイ	1 L	ア	ウ	ト ((「	マク	クロ	(]	Ma	cro)]	タ		
	ブ)	۱.																142
	タ	イム	7	ウト	• (ſ	リン	ンク	(]	Lin	ks)_	12	タブ)				143
認	識阳	夏度	(画	面	(S	cre	ens)]	タ`	ブの) [設				
(G	ene	ral)		タフ	")													143
	認	識限	度	に達	重し	た	Z	との)判	別								144
	認	識限	度	に達	達し	た	と	きの)ア	ク	ショ	ン						144

第10章 アクション、パート2:タイミ

ングの問題..............	145
アクション後の休止	. 145
アクションの処理速度	. 145
アクション間の休止 (「マクロ (Macro)」タブ)	145
休止時間の設定 (「画面 (Screens)」タブの「一	
般」タブ)..............	. 146
特定のアクションの後に休止を追加する	. 147
画面の完了	. 147
次のマクロ画面の認識が早すぎる	. 147
通常の TN3270 プロトコル	. 148
解決策	. 148
画面の完了に関係する属性	. 150

第 11 章 変数とインポートした Java

ク	ラス											-			153
変	数とイ	ンズ	パー	ト西	世の	根	要	į.							. 153
	拡張	マク	ロ形	(式)	がル	ΥĒ	要								. 153
	変数の	の有法	効範	囲											. 154
	「変数	数 (\	/ari	abl	es).		タ	ブ(の概	要					. 154
注	意を必	と要と	こす	る間	卽題	Į									. 159
	Java	ライ	ブ	ラリ	_	ま	た	は	クラ	ス	の酢	習置			. 159
	変数	名と	型名	, 1 .											. 160
	マクロ	コ間	での)変	数0)車	医说	叁.							. 160
	フィー	ール	ド変	数											. 160
変	数の値	も用													. 161
	変数7	が初	期化	:さ:	れる	5₿	寺片	₹.							. 161

標準型に属する変数の使用.......	. 161
インポート型に属する変数の使用.....	. 162
同じインポート型の変数の比較	. 163
パラメーター・リストのマクロへの引き渡し	. 164
Host On-Demand によるパラメーター・リスト	
の処理方法...............	. 165
パラメーター・リストの指定	. 165
マクロ作成者の考慮事項	. 171
チェーニングされたマクロで変数を初期化しない	
	. 171
Java メソッドの呼び出し	. 172
メソッド呼び出しを使用できる個所	. 172
メノット呼び出しの構义	. 172
マクロ・フノダイムが呼び出し先メソットを検索	170
9 ② 刀 伝	. 172
$\gamma \neq 0$ · $\gamma = \gamma + $	172
(1, 1, 2, 2, 3, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,	173
HML アドノアナー に周y Sアノアトの同じ田じ HML で開始される予約済み変数を	174
SHMI Format Utils	174
\$HMI PSU#J\$	176
\$HMI Session Util\$	182
\$HMLSOLUtils	. 183
\$HMLSystemUtil\$.	. 185
FormatNumberToString() および	
FormatStringToNumber()	. 187
, in the second s	
第 12 章 その他のセキュリティー・オ	
第 12 章 その他のセキュリティー・オ プション	189
第 12 章 その他のセキュリティー・オ プション	189 . 189
第 12 章 その他のセキュリティー・オ プション パスワードの記録 「パスワードの記録」が使用可能な場合	189 . 189 . 189
 第 12 章 その他のセキュリティー・オ プション パスワードの記録 「パスワードの記録」が使用可能な場合 「パスワードの記録」が使用不可な場合 	189 . 189 . 189 . 189 . 190
第 12 章 その他のセキュリティー・オ プション パスワードの記録 「パスワードの記録」が使用可能な場合 「パスワードの記録」が使用不可な場合	189 . 189 . 189 . 190
 第 12 章 その他のセキュリティー・オプション	189 . 189 . 189 . 190 191
 第 12 章 その他のセキュリティー・オプション	189 . 189 . 189 . 190 191 . 191
 第 12 章 その他のセキュリティー・オプション. パスワードの記録. パスワードの記録」が使用可能な場合. 「パスワードの記録」が使用不可な場合. 第 13 章 その他の拡張機能. 複数のセッションとの対話. 基本 	189 . 189 . 189 . 190 191 . 191 . 191
 第 12 章 その他のセキュリティー・オ プション	189 . 189 . 189 . 190 191 . 191 . 191 . 192
 第 12 章 その他のセキュリティー・オ プション	189 . 189 . 189 . 190 191 . 191 . 191 . 192 . 193
 第 12 章 その他のセキュリティー・オ プション	189 . 189 . 189 . 190 191 . 191 . 191 . 192 . 193 . 194
 第 12 章 その他のセキュリティー・オ プション	189 . 189 . 189 . 190 191 . 191 . 191 . 192 . 193 . 194 . 195 . 196
 第 12 章 その他のセキュリティー・オ プション	189 . 189 . 189 . 190 191 . 191 . 191 . 192 . 193 . 194 . 195 . 196 . 197
 第 12 章 その他のセキュリティー・オ プション	189 . 189 . 189 . 190 191 . 191 . 191 . 192 . 193 . 194 . 195 . 196 . 197 . 200
 第 12 章 その他のセキュリティー・オ プション	189 189 189 190 191 191 191 192 193 194 195 196 197 200 201
 第 12 章 その他のセキュリティー・オ プション	189 . 189 . 189 . 190 191 . 191 . 191 . 192 . 193 . 194 . 195 . 196 . 197 . 200 . 201
 第 12 章 その他のセキュリティー・オ プション	189 189 189 190 191 191 192 193 194 195 196 197 200 201
 第 12 章 その他のセキュリティー・オ プション	189 189 190 191 191 191 192 193 194 195 196 197 200 201 203
 第 12 章 その他のセキュリティー・オ プション	189 189 189 190 191 191 191 192 193 194 195 196 197 200 201 203
 第 12 章 その他のセキュリティー・オ プション	189 189 189 190 191 191 192 193 194 195 196 197 200 201 203 203 203
 第 12 章 その他のセキュリティー・オ プション	189 . 189 . 189 . 190 191 . 191 . 191 . 192 . 193 . 194 . 195 . 196 . 197 . 200 . 201 203 . 203 . 203 . 203
 第 12 章 その他のセキュリティー・オ プション	189 189 189 190 191 191 191 192 193 194 195 196 197 200 201 203 203 203

ストリングを指定する際のエラー.....204 コード・エディターの使用.....204 本書からコード・エディターへのスクリプトのコ ピー・アンド・ペースト.....204

第3部マクロ言語								. 207
----------	--	--	--	--	--	--	--	-------

第 15 章 マクロ言語の機能		209
XML の使用	•	209
Host On-Demand マクロ言語の XML 構文 .		209
コード・エディター		210
エレメントの階層		210
マクロ・スクリプトへのコメントの挿入		211
コメントの形式		212
コメント・エラー		212
コメントの例	•	212
- / / / / / / / / / / / / / / / / / / /	ь. К	212
へいると エレバントを使用した マノロ バノリノ	1.	212
の/ハック	1 0	213
マクロと祖の白わせた HOSt Access Toolkit 表面 使用	JU	010
(火用	•	213
第 16 音 マクロ言語エレメント	,	210
	• •	219
	•	219
XML 要件	•	219
属性値の拡張形式............	•	219
型付きデーター・・・・・・・・・・・	•	219
<actions> エレメント</actions>	•	220
概要		220
属性		220
XML サンプル		221
<attrib> エレメント</attrib>		221
概要		221
属性		221
XML サンプル		222
<pre>chorselection> エレメント</pre>	•	222
「 「 概 西 「 「 「 「 「 「 「 「 「 「 「 「 「	•	222
	•	222
商は ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	•	222
ANIL $\int \mathcal{I} \mathcal{I} \mathcal{I}$	•	222
	•	222
	•	222
	•	223
$XML \; \mathcal{T} \mathcal{T} \mathcal{T} \mathcal{T} \mathcal{T} \mathcal{T} \mathcal{T} \mathcal{T}$	•	223
<commwait> エレメント</commwait>	•	223
概要	•	223
属性	•	223
XML サンプル	•	224
<condition> エレメント</condition>		224
概要:::::::::::::::	•	224
属性::::::::::::::		224
XML サンプル		224
<create> エレメント</create>		225
概要		225
属性		225
XML サンプル		225
<cursor> エレメント</cursor>		226
概要		226
属性	•	226
//***:	•	226
<pre>////////////////////////////////////</pre>	•	220
(USIOND エレハント	•	227
1943	•	<i>∠∠1</i>

属性......							. 227
XML サンプル							. 227
<customreco> エレメント</customreco>							. 227
概要							. 227
属性							. 228
XML サンプル	•••		•	•	•	•	228
<pre>cdescription> TLXY</pre>	•••	• •	•	•	•	•	228
概要	•••	• •	•	•	•	•	. 220
	• •	• •	•	·	·	•	. 220
	•••	• •	•	•	•	•	. 229
$\operatorname{AML} \operatorname{y}_{\mathcal{F}} \operatorname{y}_{\mathcal{F}} \operatorname{y}_{\mathcal{F}}$		• •	·	·	·	•	. 229
	• •	• •	·	·	·	·	. 229
概要		• •	•	•	·	•	. 229
属性	• •	• •	•	·	·	·	. 229
XML サンブル			•	•	•	•	. 229
<extract> エレメント</extract>						•	. 230
概要							. 230
属性							. 230
XML サンプル							. 231
<fileupload> エレメント</fileupload>							. 231
概要							. 231
属性			-	-	-	-	231
MI サンプル	•••	• •	•	•	•	•	233
filevfors TLXY	• •	• •	•	•	·	•	. 200
	•••	• •	•	•	•	•	. 233
		• •	·	·	·	•	. 233
周任	• •	• •	•	·	·	·	. 233
XML サングル	•••	• •	·	·	·	·	. 234
<hascript> エレメント.</hascript>			•	•	•	•	. 234
概要			•	•	•	•	. 234
属性......						•	. 234
XML サンプル							. 236
<if> エレメント</if>							. 237
概要							. 237
属性......							. 237
XML サンプル							. 237
<import> エレメント</import>							238
概要	•••	• •	•	•	•	•	238
属性	•••	• •	•	•	•	•	. 200
府正 · · · · · · · · · · · · · · · · · · ·	• •	• •	•	·	•	•	. 230
	• •	• •	·	·	·	·	. 200
<input/> エレメント	• •	• •	·	•	·	·	. 239
概要		• •	•	•	·	•	. 239
属性	• •	• •	•	·	·	•	. 239
XML サンブル		• •	•	•	·	•	. 240
<message> エレメント .</message>				•	•	•	. 240
概要							. 240
属性......							. 240
XML サンプル							. 240
<mouseclick> エレメント</mouseclick>							. 240
概要							. 240
属性							. 240
XML サンプル			•				. 241
<nextscreen> エレメント</nextscreen>			•	•	•	•	. 241
概要	• •	• •	•	•	·	·	· 211
144.女 • • • • • • • • • • • • • • • • • • •	•••	• •	•	•	·	·	. 241 0/1
時止・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・		• •	•	•	·	·	. 241
$\operatorname{AML} \mathcal{Y} \neq \mathcal{I} \mathcal{W} \cdot \cdot \cdot \cdot$	• •	• •	·	·	·	·	. 241
<nextscreens> エレメント</nextscreens>	• •	• •	•	•	·	·	. 241
概要						•	. 241

. 227	属性
. 227	XML サンプル
. 227	<numfields> エレメント</numfields>
. 227	概要
. 228	属性
. 228	XML サンプル
. 228	<numinputfields> エレメント</numinputfields>
. 228	概要
. 229	属性
. 229	XML サンプル
. 229	<oia> エレメント 243</oia>
229	概要 243
229	属性 244
229	XMI + ンプル 244
230	AME / / / / · · · · · · · · · · · · · · ·
230	「如山」ション・ション・ション・ション・ション・ション・ション・ション・ション・ション・
. 230	碱女····································
. 230	内止 ・・・・・・・・・・・・・・・・・244 VMI サンプル 245
. 231	$AIVIL y \neq y \neq y = 0$
. 231	<pre>vpenonits エレメント・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・</pre>
. 231	
. 231	周性
. 233	$\operatorname{XML} \mathcal{I} \mathcal{I} \mathcal{I} \mathcal{I} \mathcal{I}$
. 233	$\langle playmacro \rangle \perp \nu \rangle \rangle \land \dots \dots$
. 233	
. 233	
. 234	$XML \forall \mathcal{Y} \mathcal{Y} \mathcal{V} \dots \dots$
. 234	$<$ print> $\pm \nu \times \nu$ +
. 234	概要
. 234	属性
. 236	XML サンブル
. 237	<pre><pre>prompt> エレメント</pre></pre>
. 237	概要
. 237	属性
. 237	XML サンブル
. 238	<recolimit> エレメント</recolimit>
. 238	概要
. 238	属性
. 238	XML サンプル
. 239	<runprogram> エレメント</runprogram>
. 239	概要
. 239	属性
. 240	XML サンプル
. 240	<screen> エレメント</screen>
. 240	概要
. 240	属性
. 240	XML サンプル
. 240	<sqlquery> エレメント</sqlquery>
. 240	概要
. 240	属性
. 241	XML サンプル
. 241	<pre><string> エレメント</string></pre>
. 241	概要
. 241	属性
. 241	XML サンプル
. 241	<trace> エレメント</trace>
. 241	概要

vi IBM Host On-Demand バージョン 13.0: Host On-Demand マクロ・プログラミング・ガイド

	属性											256
	XML	サ	ンフ	プル								256
<t< td=""><td>ype></td><td>エレ</td><td>ノメ</td><td>ン</td><td>arepsilon</td><td></td><td></td><td></td><td></td><td></td><td></td><td>257</td></t<>	ype>	エレ	ノメ	ン	arepsilon							257
	概要											257
	属性											257
	XML	サ	ンフ	プル								257
<v< td=""><td>vars></td><td>エレ</td><td>ノメ</td><td>ント</td><td>\vdash</td><td></td><td></td><td></td><td></td><td></td><td></td><td>257</td></v<>	vars>	エレ	ノメ	ント	\vdash							257
	概要											257
	属性											258
	XML	サ	ンフ	プル								258
<1	arupd	late	> -	エレ	ノメ	ン	ト					258
	概要											258
	属性											258
	XML	サ	ンフ	プル								259

第 **17**章 サンプル・マクロ・コード **261** Excel スプレッドシートまたは DB2 データベース への CICS トランザクション・レコードのコピー . 261

概要	. 261 . 261 . 264
付録 A. 追加情報	267
オルト結合規則	. 267
規則の記述	. 267
入力アクションの略号キーワード......	. 267
付録 B. 特記事項	271
付録 C. 商標	273

viii IBM Host On-Demand バージョン 13.0: Host On-Demand マクロ・プログラミング・ガイド

本書について

この「マクロ・プログラミング・ガイド」は、より良い Host On-Demand マクロ を作成するのに役立ちます。このガイドは、Host On-Demand マクロの開発者およ び通常のユーザーを対象にしています。次の 3 つのパートがあります。

1 ページの『第1部マクロの基本』では、基本的な概念を説明し、ツールを紹介し、単純マクロの記録と再生の方法を手順を追って説明します。

31 ページの『第 2 部 マクロの開発』では、Host On-Demand マクロ・システムの機能を詳しく説明します。

207 ページの『第 3 部 マクロ言語』では、XML マクロ言語について説明します。

「マクロ・プログラミング・ガイド」 は Host On-Demand オンライン Knowledge Center (http://www-01.ibm.com/support/knowledgecenter/ SSS9FA_13.0.0/com.ibm.hod.doc/doc/macro/macro.html) にあります。

MySupport 機能を使用すると、サポート・ビューを個人別に設定できます。また、 IBM 製品に関する新しいパッチ、ダウンロード、およびその他のテクニカル・サポ ート情報を掲載した電子メール通知を毎週受け取るための登録を行うことができま す。 MySupport に登録するには、URL http://www.ibm.com/support/ docview.wss?uid=swg21168680 にある技術情報の説明に従ってください。

その他の Host On-Demand 資料

この「マクロ・プログラミング・ガイド」の他に、Host On-Demand には、本製品 の使用に役立つその他の情報源も用意されています。ここに記載する資料にアクセ スするには、Host On-Demand ライブラリー・ページ (http://www.ibm.com/ software/webservers/hostondemand/library.html) にアクセスしてください。大部 分の資料は、Host On-Demand 製品、または Toolkit CD-ROM にも含まれていま す。

- オンライン・ヘルプ。オンライン・ヘルプは、Host On-Demand のインストー ル完了後に管理者やユーザーが利用できる基本情報源です。オンライン・ヘルプ は、Host On-Demand タスクの実行方法について、詳しい手順を提示します。 目次と索引は、タスク指向のヘルプ・パネルや概念ヘルプ・パネルを見付けるの に役立ちます。 Host On-Demand のグラフィカル・ユーザー・インターフェー ス (GUI) を使用しているときは、「ヘルプ」ボタンをクリックすると、GUI 用 のパネル・レベルのヘルプ・パネルが立ち上がります。
- Programming, Installing, and Configuring Host On-Demand。この資料は、システム管理者を対象にし、Host On-Demand プログラムの計画、インストール、および構成に役立ちます。
- Program Directory。このプログラム・ディレクトリーは、z/OS および OS/390 プラットフォームに Host On-Demand をインストールする方法について説明し ます。

- *Readme* ファイル。この readme.html ファイルには、製品資料に記載するのに間 に合わなかった最新の製品情報が記載されています。
- Web Express Logon 解説書。この解説書は、Web Express Logon の理解、イン プリメンテーション、およびトラブルシューティングの方法を段階的に説明して います。Web Express Logon の概要、ご使用の環境におけるWeb Express Logon の計画とデプロイに役立つ 2 つのシナリオ・ベースの例、およびカスタ マイズされたマクロとプラグインの作成用の複数のAPI を記載しています。
- ホスト印刷解説書。ホスト・セッションを構成した後、ユーザーがローカルまたは LAN 接続のプリンターやファイルに、ホスト・セッション情報を印刷できるようにする場合に、この「ホスト印刷解説書」を使用します。
- Session Manager API 解説書。この解説書は、ホスト・セッション、およびホスト・セッションとのテキスト・ベースの対話を管理するための JavaScript API について記載しています。
- Programmable Host On-Demand 解説書。この解説書は、開発者が各種 Host On-Demand クライアント・コード (例えば、端末、メニュー、およびツールバ ー)を独自のカスタム Java アプリケーションとアプレットに統合できるように する、1 組の Java API について説明しています。
- Toolkit 概説。この資料は、Host On-Demand Toolkit のインストールと構成の 方法を説明します。この Toolkit は、Host Access Client Package に付属して いますが、Host On-Demand 基本製品とは別の CD-ROM からインストールさ れます。 Host On-Demand Toolkit は、ご使用の環境で Host On-Demand を 最大限に活用するのに役立つ Java Bean やその他のコンポーネントを提供する ことによって、Host On-Demand 基本製品を補完します。
- Host Access Beans for Java Reference。この資料は、Host On-Demand Toolkit に含まれています。 Java Bean を使用して Host On-Demand 環境をカスタマ イズし、エミュレーター・セッションのステップを自動化するマクロを作成した いプログラマーの解説書の役目をします。
- ホスト・アクセス・クラス・ライブラリー解説書。この資料は、Host On-Demand Toolkit に含まれています。データ・ストリーム・レベルでホスト 情報にアクセスできる Java アプレットやアプリケーションを作成したいプログ ラマーの解説書の役目をします。
- J2EE Connector Reference。この資料は、Host On-Demand Toolkit に含まれています。 Java 2 Enterprise Edition (J2EE) 互換アプリケーションにアクセスするアプレットやサーブレットを作成したいプログラマーの解説書の役目をします。
- Host On-Demand Redbooks。 Host On-Demand レッドブックは、Host On-Demand を使用する実践的な方法を提示することによって、Host On-Demand 製品資料を補完します。レッドブックは、「現状のまま」提供され ているので、常に最新の製品情報が記載されているとは限りません。すべての Host On-Demand レッドブックの最新のリストについては、次の Host On-Demand ライブラリー・ページにアクセスしてください。 http://www.ibm.com/software/webservers/hostondemand/library.html

本書の規則	
	この「マクロ・プログラミング・ガイド」では、次の表記規則を使用しています。
表 1. 本書の規則	
規則	意味
モノスペース	コマンド・プロンプトで入力する必要のあるテキスト、およびコマンド、関数、ならび にリソース定義属性やそれらの値など、示されているとおりに使用しなければならない 値を示します。モノスペースは、画面テキストおよびコード例を示すためにも使用され ます。
イタリック	指定する必要のある可変値を示します (例 : file_name にファイルの名前を指定しま す)。イタリックは、強調表示および書名の表示にも使用されます。
Return	Return、Enter または左矢印が表示されているキーを指します。
>	メニューの記述に使用している場合は、一連のメニュー選択を示します。例えば、 「「ファイル」>「新規」をクリックします」とある場合は、「「ファイル」メニュー から「新規」コマンドをクリックします」という意味になります。
	ッリー表示の記述に使用している場合は、一連のフォルダーまたはオブジェクトの展開 を示します。例えば、「HODConfig Servlet > Sysplexes > Plex1 > J2EE Servers > BBOARS2 の順に展開する」とある場合は、次の意味になります。
	1. HODConfig Servlet フォルダーを展開する
	2. Sysplexes フォルダーを展開する
	3. Plex1 フォルダーを展開する
	4. J2EE Servers フォルダーを展開する
	5. BBOARS2 フォルダーを展開する
Java	本書では、Java は、1.6 およびそれ以降の JVM で実装されていることを意味します。



この図は、読者に対する注を強調表示するのに使用します。

この図は、読者に対するヒントを強調表示するのに使用します。

xii IBM Host On-Demand バージョン 13.0: Host On-Demand マクロ・プログラミング・ガイド

第1部マクロの基本

2 IBM Host On-Demand バージョン 13.0: Host On-Demand マクロ・プログラミング・ガイド

第1章概要

Host On-Demand マクロ

マクロの定義

Host On-Demand マクロは、Host On-Demand クライアントが、端末エミュレー ター・セッション (3270 ディスプレイ・セッション、5250 ディスプレイ・セッシ ョン、VT ディスプレイ・セッション、または CICS ゲートウェイ・セッション) で実行されるホスト・アプリケーションと自動的に対話できるようにする XML ス クリプトです。 Host On-Demand マクロは、通常、単一のホスト・アプリケーシ ョンを伴う特定のタスク、または一連のタスクを実行するために作成されます。

マクロの利点

人間のオペレーターと比較すると、マクロは、より迅速に、かつよりエラーのリス クが少ない方法で、ホスト・アプリケーションと対話します。マクロでは、そのマ クロを実行する人物が、ホスト・アプリケーションの操作方法を知っている必要は ありません。多くの場合、マクロは自動で実行されます。マクロは繰り返し使用す ることができます。マクロをコピーし、複数のユーザーに配布することができま す。しかも、最も重要なことは、マクロが、ホスト・アプリケーションとワークス テーション・アプリケーションの両方を必要とするカスタマー要件に対して、適用 範囲が広い解決法の一部として役に立つ点です。

初級ユーザー

初級ユーザーは、ホスト・アクセスとの単調な時間のかかる対話を自動化するため の基本的なマクロを作成できます。ユーザーは、ホスト・アプリケーションをナビ ゲートして、最初に見たい画面に進み、「マクロを記録 (Record Macro)」アイコン を選択し、ホスト・アプリケーションを使用してタスクを実行します。ユーザーが アクセスするアプリケーション画面ごとに、Host On-Demand は、アプリケーショ ン画面のイメージ、およびアプリケーション画面へのユーザーの入力を記録しま す。ユーザーが、以前と同じアプリケーション画面から始まる記録済みマクロを再 生すると、Host On-Demand は、以前に記録されたイメージに基づいて、各ホス ト・アプリケーション画面を認識し、人間のオペレーターが以前に実行したアクシ ョンを繰り返します。

上級ユーザー

上級のユーザーは、ホスト・アクセス・マクロ・エディター (マクロ・エディター) を使用して、記録されたマクロを追加または改善することができます。このツール は、セッション・パネル上のアイコンをクリックすることによって選択可能です。 このツールが提供するグラフィカル・ユーザー・インターフェース (入力フィール ド、テキスト・ボックス、チェック・ボックスなどから構成されます) では、ユー ザーがホスト・アプリケーションとの各画面の対話を変更したり、機能を追加した りすることができます。ユーザーがマクロの画面認識動作とユーザー入力を編集 し、拡張できるようにする機能に加えて、マクロ・エディターは、アプリケーショ ンを通じた代替パスの選択、処理する必要がない画面のスキップ、または直前の画 面に戻るなどのより高度な動作を、ユーザーがマクロに追加できるようにする機能 も備えています。さらに、セッション画面からデータを読み取り、処理する機能、 オペレーターに状況を通知する機能、重要な決定をオペレーターに促す機能、ホス トからファイルをダウンロードまたはアップロードする機能、アプリケーション画 面の印刷を自動化する機能を含めて、さらに強力な機能もあります。

プログラミング機能

マクロ・エディターは、プログラミング機能も備えています。プログラミングの経 験があるユーザーは、Host On-Demand マクロに機能を追加することができます。 これを行うには、変数の作成と操作、演算式と論理式の使用、if-then-else 条件の作 成、別のマクロへのチェーニング、外部 Java ライブラリーに保管されている Java メソッドの呼び出し、ネイティブ・アプリケーションの起動、およびトレース情報 の書き込みを行います。

コード・エディターは、マクロ・エディターから立ち上げるテキスト・エディター です。コード・エディターを使用すると、ユーザーは、マクロ・スクリプトを構成 する XML エレメントを表示し、変更したり、システムのクリップボードを使用し てテキストをカット・アンド・ペーストしたりすることができます。

サンプル

本書には、マクロ・コードのサンプルが記載されています。 261 ページの『第 17 章 サンプル・マクロ・コード』には、サンプル CICS データベースから項目を読み 取り、Microsoft Excel スプレッドシートに書き込むマクロの例が記載されていま す。

マクロの配置

Host On-Demand デプロイメント・ウィザードに組み込まれている設定値を使用す ると、システム管理者は、Web ロケーション、もしくは LAN または DVD-ROM ドライブに置かれているサーバー・ライブラリーでユーザーにマクロを配置するこ とができます。

詳しくは、「Host On-Demand 計画、インストール、および構成」の『サーバー・ マクロ・ライブラリーの作成とデプロイ (Creating and deploying server macro libraries)』を参照してください。

ローカル・ユーザーは、書き込みアクセス権のある場所であればどこでにでもマク ロを保管できます。

マクロを使用したエンタープライズ・アプリケーションの統合

Host On-Demand マクロを使用すると、Telnet アクセス可能なアプリケーション を、ワークステーション・アプリケーションに統合することができます。マクロ は、Telnet アクセス可能なアプリケーションとの間でデータが出入りするパスを提 供します。

Host On-Demand には、マクロを駆動できるようにする次の 2 つのプログラミン グ・インターフェースを組み込んでいます。 • Programmable Host On-Demand

これは、開発者が各種 Host On-Demand クライアント・コード (例えば、端 末、メニュー、およびツールバー) を独自のカスタム Java アプリケーションと アプレットに統合できるようにする、1 組の Java API です。

詳しくは、Host On-Demand 資料の「*Programmable Host On-Demand*」を参照 してください。

• Session Manager API

Session Manager API は、ホスト・セッション、およびホスト・セッションとの テキスト・ベースの対話を管理するための JavaScript API です。

詳しくは、Host On-Demand 資料の「Session Manager API 解説書」を参照し てください。

Host Access Toolkit

Host Access Toolkit は、Host On-Demand クライアントのプログラマチック制御 とその他の機能を提供する別個の製品であり、Host On-Demand マクロを起動し、 対話するための Java API を組み込んでいます。

マクロとセキュリティー

マクロは、ユーザーとホスト・アプリケーション間の対話を簡単にトランスポート できる、暗号化されていないテキスト・ベースで表記したものであるため、貴重な 知的資産としてマクロの保護を検討する必要があります。

特に、暗号化されていないパスワードやその他の機密データをマクロ・スクリプト に保管しないようにする必要があります。代わりに、次のようにすることができま す。

- 例えば、パスワードの入力をユーザーに求めたり、ホストまたはローカル・アプ リケーションからデータを取得したりするなど、外部ソースから機密情報を取得 するようにマクロを設計する。
- 以下の機能を使用して、入力を暗号化する。
 - FileUpload アクションの「パスワード (Password)」フィールド (95 ページ の『ユーザー ID とパスワード』を参照)。
 - 入力アクションの「パスワード (Password)」チェック・ボックス (99 ページの『パスワード』を参照)。
 - プロンプト・アクションの「パスワードの応答」リスト・ボックス (116 ペ ージの『パスワード応答』を参照)。
 - SQLQuery アクションの「パスワード」フィールド(125 ページの『ユーザ ー ID とパスワード』を参照)。
 - 「パスワードを記録」オプション (デフォルトで使用可能) (189 ページの 『パスワードの記録』を参照)。

本書における 3270 アプリケーション

マクロは、3270 ディスプレイ・セッション、5250 ディスプレイ・セッション、VT ディスプレイ・セッション、および CICS ゲートウェイ・セッションで使用できま すが、本書では、3270 ディスプレイ・セッションと 3270 ホスト・アプリケーショ ンを中心に論じています。

第2章 マクロのコンポーネント

概要

この章では、Host On-Demand のマクロ機能を使用するときに出てくるコンポーネ ントについて説明します。主なコンポーネントの概要は次のとおりです。

- マクロ・マネージャー。これは、マクロの処理を可能にする、Host On-Demand 内のすべてのユーザー・インターフェースを指す広義語です。マクロ・マネージ ャーは、次の3つのメイン・ユーザー・インターフェースから構成されていま す。
 - マクロ・マネージャー・ツールバー。このツールバーは、よく使用されるマクロ機能(例えば、「マクロを記録(Record a Macro)」、「マクロを再生(Play Macro)」、「マクロのプロパティーを編集(Edit Macro Properties)」など)のアイコンを表示します。この章の『マクロ・マネージャー・ツールバー』を参照してください。
 - マクロ・エディター。マクロ・スクリプトを編集するためのメイン・ユーザ
 ー・インターフェースです。この章の 9 ページの『マクロ・エディター』
 を参照してください。
 - コード・エディター。マクロ・スクリプトの XML 言語を直接編集できるようにするテキスト・エディターです。この章の 10 ページの『コード・エディター』を参照してください。
- マクロ・ランタイム。マクロを再生するプログラム・モジュールです。 11 ペ ージの『マクロ・ランタイム』を参照してください。
- マクロ・オブジェクト。マクロ機能を提供する、基礎の Java オブジェクトです。 11 ページの『マクロ・オブジェクト』を参照してください。

この章の後続のセクションでは、これらのコンポーネントについて詳しく説明しま す。この章の最後のセクションでは、本書で使用されるその他のマクロ用語を定義 します。

マクロ・マネージャー

マクロ・マネージャーは、すべてのマクロ・ユーザー・インターフェースを表す包括的な用語です。 3 つのメイン・インターフェース、すなわちマクロ・マネージャ ー・ツールバー、マクロ・エディター、およびコード・エディターがあります。

マクロ・マネージャー・ツールバー

マクロ・マネージャー・ツールバーには、一般的なマクロ操作用のアイコンが含ま れています。これらのアイコンはすべて、マクロ開発者によって使用されますが、 少なくとも 1 つのアイコン (「マクロを再生 (Play macro)」アイコン) は、マクロ のユーザー用でもあります。 8 ページの図 1 は、マクロ・マネージャー・ツール バーを示しています。 (図示するために、この図はある意味では不正確です。この 図は、ツールバー上のすべてのボタンが同時に使用可能になっている状態を示して

ispf_usp.mac	- 4	×	e 📕	**	Ø 1	
図 1. マクロ・マネージャー・ツールバ	<u></u>	010042000				000000

マクロ・マネージャー・ツールバーを表示する手順は、次のとおりです。

- 1. Host On-Demand クライアントを開始する。
- 端末エミュレーター・セッション (3270 ディスプレイ・セッション、5250 ディ スプレイ・セッション、VT セッション、または CICS ゲートウェイ・セッシ ョン) を開始する。
- 3. 「表示 (View)」 > 「マクロ・マネージャー」の順に選択する。

企業のディスプレイ・セッションの構成によっては、マクロ・マネージャー・ツー ルバー上のアイコンの一部がメイン・ツールバー上にも表示される場合がありま す。この配置は、ユーザーの便宜のためであり、問題にはなりません。アイコンが どのツールバーに表示されても、機能は同じです。

マクロ・マネージャー・ツールバーの各部の機能を、左から右の順に、以下に簡単 に説明します。これらの機能の詳細については、本書の後の部分で説明します。

- 現在選択されているマクロ。ツールバーの左側にあるこの白いテキスト・フィー ルドは、入力フィールドではなく、現在選択されているマクロの名前をマクロ・ マネージャーが表示するテキスト・フィールドです。ここで現在選択されている マクロは ispf_usp.mac です。
- 「マクロを選択 (Select a macro)」。このアイコンは、大きな下向きの矢印です。このアイコンをクリックすると、再生、編集、コピー、または削除のために現行のマクロを選択できます。
- 「現行マクロのプロパティーを編集 (Edit current macro properties)」。このア イコンをクリックすると、マクロ・エディターが立ち上がります (9 ページの 『マクロ・エディター』を参照)。
- 「現行マクロをリストから削除 (Delete current macro from list)」。このアイ コンをクリックすると、現在選択されているマクロを削除します。
- 「マクロを再生 (Play macro)」。このアイコンをクリックすると、現在選択されているマクロを再生します。
- 「マクロを記録 (Record macro)」。このアイコンをクリックすると、新しいマ クロを記録します。
- 「マクロの再生または記録を停止 (Stop playing or recording macro)」。この アイコンをクリックすると、マクロの再生または記録を終了します。
- 「マクロの再生または記録を一時停止 (Pause playing or recording macro)」。
 このアイコンをクリックすると、マクロの再生または記録を一時的に停止します。
- 「記録を追加 (Append recording)」。このアイコンをクリックすると、記録対象の現在のソースとしてアクティブ・セッションが選択されます (197 ページの 『複数のセッションと対話するマクロの記録』を参照)。
- 「プロンプトを追加 (Add a prompt)」。

- 「スマート待機時間を追加 (Add a Smart Wait)」。
- 「抽出を追加 (Add an Extraction)」。

単純マクロの記録と再生のプロセスを段階的にたどるには、 13 ページの『第 3 章 単純マクロの記録と再生』を参照してください。

マクロ・エディター

マクロ・エディター (フルネームはホスト・アクセス・マクロ・エディター) は、マ クロの各部を編集するためのグラフィカル・ユーザー・インターフェース (ボタ ン、入力フィールド、リスト・ボックスなどを備えた) です。図 2 は、マクロ・エ ディターを示しています。

Host Access Macro Editor	×
Macro: Screens Links Variables	
Means Name listf av1	
Author	
Creation Date	
✓ Pause Between Actions 300 milliseconds	
✓ Timeout Between Screens 60000 milliseconds	
Show All Prompts at Start of Macro	
Use Variables and Arithmetic Expressions in Macro	
Save and Exit Save As Save Cancel Code Editor Import Export Help	
Define the general attributes of the macro	

図 2. マクロ・エディター

おそらく、大部分のマクロ作成にマクロ・エディターを使用することになります。 次のサブセクションで説明しているコード・エディターよりも強力 (データの管理 が簡単という点で)ですが、コード・エディターの機能をすべて実行できるわけで はありません。

マクロ・エディターを起動するには、マクロ・マネージャー・ツールバーに進みます。

- 1. 「マクロを選択 (Select a macro)」アイコンをクリックして、編集するマクロを 選択する。
- 2. 「現行マクロのプロパティーを編集 (Edit current macro properties)」アイコン をクリックして、マクロ・エディターでマクロを編集する。

コード・エディター

マクロ・エディターとコード・エディターはどちらも、マクロを編集します。どち らのツールも、同じタイプの基礎マクロ・ソース、つまり XML スクリプトとの間 で読み書きします。ただし、それぞれのツールには、適したタスクがあります。

マクロ・エディターは、強力で使いやすいグラフィカル・ユーザー・インターフェ ースを備え、マクロの作成と編集に優れたツールです。

一方、コード・エディターは、マクロが構成される XML エレメントを直接編集で きる、テキスト・エディター・インターフェースを備えています。図 3 は、マク ロ・スクリプトを表示するコード・エディターを示しています。

Code Editor
<pre><hascript description=" " name="ispf_usp" pausetime="300" pre="" prom<="" timeout="60000"></hascript></pre>
<pre><screen entryscreen="true" exitscreen="false" name="Screen1" transient+"false<br=""><description> <oia invermatch="false" optional+"false"="" status="NOTINHIBITED"></oia> </description> <actions> <mouseclick col="15" row="4"></mouseclick> <mouseclick col="15" row="4"></mouseclick> <input 0"="" col="0" movecursor="true" row="0" value="3(enter)" xlatehostkeys="
</actions>
<nextscreens timeout="/> <nextscreen name="Screen2"></nextscreen> </actions></screen></pre>
<pre><screen <="" entryscreen="false" exitscreen="true" name="Screen2" td="" transient="false"></screen></pre>
Messages
OK Verify キャンセル

図 3. コード・エディター

コード・エディターは、次のような専門性の高い編集タスクに使用してください。

- マクロ・エディターがアクセスできない XML エレメントのいくつかの属性を変 更する。
- XML エレメントを調べて、マクロの動作を理解していることを確認する。
- マクロをデバッグする。
- Windows のクリップボードを使用して、XML コードを別のソースからカット・アンド・ペーストする。

コード・エディターを立ち上げる手順は、次のとおりです。

- 1. マクロ・エディターを使用して、編集したいファイルを開く。
- 2. マクロ・エディター・ウィンドウの下部にあるボタン行で、コード・エディター をクリックする。

マクロ・ランタイム

マクロ・ランタイムは、ユーザーが「マクロを再生 (Play Macro)」アイコンをクリ ックするとマクロを再生するプログラム・モジュールです。具体的には、マクロ・ ランタイムは、現行のマクロ・スクリプトの内容を読み取り、マクロの再生を生成 します。

マクロ・オブジェクト

マクロ・オブジェクトは、マクロ・マネージャー・ツールバー、マクロ・エディタ ー、コード・エディター、およびマクロ・ランタイムの基盤を成す機能を提供す る、Java インスタンスです。

IBM Host Access Toolkit (別途購入する製品) は、マクロ・クラスの複数のメソッ ドを介して、マクロ・オブジェクトへのプログラミング・アクセスを提供します。 本書では、Host Access Toolkit の使用方法を説明しません。

その他の用語の定義

本書で使用されるその他の用語の定義は、次のとおりです。

表 2. 用語の定義

アクション	アクションは、マクロを再生するときにマクロ・ランタイムが実
	行するアクティビティー (セッション・ウィンドウへの一連のキ
	ーの送信、ポップアップ・ウィンドウでのプロンプトの表示、画
	面からのテキスト・ブロックの取り込みなど)を指定する命令で
	す。アクションの編集または作成は、マクロ・エディターで行う
	ことができます。個々のアクション・エレメントの表示と変更
	は、コード・エディターで行います。 77 ページの『第8章マ
	クロ・アクション』を参照してください。
アプリケーション画面	アプリケーション画面は、ホスト・アプリケーションによって
	Host On-Demand セッション・ウィンドウ上に表示される、意味
	のある文字配置です。 24 ページの『アプリケーション画面』を
	参照してください。
ディスクリプター	ディスクリプターは、アプリケーション画面の 1 つの特性を記述
	する命令です。ディスクリプターの編集または作成は、マクロ・
	エディターで行うことができます。個々のディスクリプター・エ
	レメントの表示と変更は、コード・エディターで行います。 54
	ページの『記述タブの概要』を参照してください。
マクロ画面	マクロ画面は、特定のアプリケーション画面の個々のアクセスを
	管理する方法をマクロ・ランタイムに指示する、1 組の命令で
	す。 25 ページの『マクロ画面』を参照してください。
マクロ・スクリプト	マクロ・スクリプトは、マクロが保管される先の XML スクリプ
	トです。マクロ・スクリプトの編集は、コード・エディターを使
	用して直接行うか、マクロ・エディターを使用して間接的に行う
	ことができます。マクロを再生すると、マクロ・ランタイムがス
	クリプト内の命令を実行します。 19 ページの『マクロ・スクリ
	プト』を参照してください。

表 2. 用語の定義 (続き)

有効な次画面	有効な次画面は、マクロの再生時に処理対象の次のマクロ画面に
	なる、有効なマクロ画面候補です。 44 ページの『ステージ 1
	の詳細』を参照してください。

第3章 単純マクロの記録と再生

この章では、次の 3 つの基本タスクを手順を追って説明することによって、マク ロ・マネージャーを実践的に紹介します。

- 単純マクロを記録する
- 記録されたマクロを再生する
- マクロの再生を特定のキーの組み合わせに割り当てる

これと同じ手順を自分で実行するには、Host On-Demand 3270 ディスプレイ・セッションを開始し、MVS システムに接続し、TSO にログオンします。ログオンすると、最初に「ISPF 基本オプション・メニュー (ISPF Primary Option Menu)」アプリケーション画面が表示されます (14 ページの図 5 を参照)。

これを実行する前に、必ずシステム管理者の許可を受けてください。必要に応じて、経験のある ISPF ユーザーに付き添ってもらってください。

単純マクロの記録

ここでは、非常に単純なマクロの記録方法を示します。このマクロは、アプリケー ション画面を「ISPF 基本オプション・メニュー (ISPF Primary Option Menu)」か ら、「ユーティリティー選択パネル (Utility Selection Panel)」画面を経由して、 「データ・セット・リスト・ユーティリティー (Data Set List Utility)」画面に変え ます。

このマクロを記録する場合は、事前に次のことを確認してください。

- 「ISPF 基本オプション・メニュー (ISPF Primary Option Menu)」が、セッション・ウィンドウに表示されている。 14 ページの図 5 を参照。
- このセッション・ウィンドウの上に、マクロ・マネージャー・ツールバーが表示 されている。図 4 は、マクロ・マネージャー・ツールバーを示しています (これ は、8ページの図 1 に表示されている図と同じです)。

ispf_usp.mac 🔻 💰 👗 📥 👑 👹 🖷 💮 💼

図 4. マクロ・マネージャー・ツールバー

システム上にマクロ・マネージャー・ツールバーが表示されない場合は、「表示 (View)」> 「マクロ・マネージャー」の順にクリックしてください。

マクロ・マネージャー・ツールバーの詳細については、 7 ページの『マクロ・マネ ージャー・ツールバー』を参照してください。

マクロを記録する手順は、次のとおりです。

1. TSO が ISPF 基本オプション画面を表示していますか。表示していない場合 は、ISPF 基本オプション画面に進みます。 14 ページの図 5 を参照。

- 2. 「マクロを記録 (Record macro)」アイコンをクリックして、記録を開始する (このアイコンは、カセットの画像上に 1 つのドットを表示しています)。
- 3. 「マクロを記録 (Record Macro)」ウィンドウが表示されます。次の手順を実 行してください。
 - a. マクロの場所、例えばパーソナル・ライブラリーを選択する。
 - b. 「新規 (New)」をクリックする。
 - c. 「名前」フィールドに名前 (例えば、ispf_ex1) を入力する (マクロ名は 大/小文字が区別されます)。
 - d. 「記述 (Description)」フィールドに説明 (例えば、Simple macro) を入力 する。
 - e. 「OK」をクリックする。
 - f. 「マクロを記録」ウィンドウが閉じます。
- ISPF 基本オプション・メニュー (ISPF Primary Option Menu)」が表示されたままです。 図 5 を参照。

3270 Diplay - A - TN03001
<u>File Edit View Communication Actions H</u> elp
= =
ispf_ex1
<u>M</u> enu <u>U</u> tilities <u>C</u> ompilers <u>O</u> ptions <u>S</u> tatus <u>H</u> elp
ISPF Primary Option Menu Option ===> _
0Settings ViewTerminal and user parameters pisplay source data or listingsUser ID.::User1 Time1ViewDisplay source data or listingsTerminal.::32782EditCreate or change source dataScreen::13UtilitiesPerform utility functionsScreen:14ForegroundInteractive language processing EachAppl ID:ISR5BatchSubmit job for language processing CommandFnter TSO or Workstation commands System ID.:MVS059 MVS acct.:7Dialog TestPerform dialog testing IBM ProductsLibrary administrator functions SW Configuration Library ManagerMVS059 MVS acct.:System ID.::11WorkplaceISPF Object/Action Workplace P anel 2Panel 2 Panel 2Panel 2Panel 2VPanel 3Panel 3(even more options)Screen options)
Enter X to Terminate using log/list defaults F1=Help F2=Split F3=Exit F7=Backward F8=Forward F9=Swap F10=Actions F12=Cancel
Recording macro

図 5. 「ISPF 基本オプション・メニュー (ISPF Primary Option Menu)」

上記の図では、Host On-Demand はアプリケーション画面を表示し、いつも と同じようにユーザーの入力を待機しています。しかし、同時に、マクロ・オ ブジェクトもユーザーの入力を記録するために待機しています。上記の図に は、マクロが記録中であることを示す、次の 2 つの視覚的合図があります。

 セッション・パネルの下部にある状況表示行に、「マクロを記録中 (Recording macro)」というメッセージが表示される。

- マクロ・マネージャー・ツールバーで、右側の5つのアイコン(「マクロを停止」、「マクロを一時停止」、および3つの「追加」アイコン)が使用可能であり、左側の5つのアイコンが一時的に使用不可である。14ページの図5を参照。
- アプリケーション画面の一番上の近くにある「オプション (Option)」行をクリ ックする。オプション行は、「ISPF 基本オプション・メニュー (ISPF Primary Option Menu)」の 4 行目であり、画面の左端の Option ===> というラベル から始まります。クリックした場所にテキスト・カーソルが表示されていなけ ればなりません。カーソルがオプション行に表示されますか。表示されない場 合は、もう一度クリックしてください。
- 6. 3 と入力し、Enter キーを押す。上記の図に表示されているように、3 は Utilities を選択します。
- 7. アプリケーション画面が、「ユーティリティー選択パネル (Utility Selection Panel)」に変わります。図 6 を参照してください。

3270 Diplay - A - TN03001			
<u>File Edit View Communication Actions Help</u>			
S S S S S S S S S S S S S S S S S S S			
ispf_ex1			
<u>M</u> enu <u>H</u> elp			
Option ===>			
1 Library Compress or print data set. Print index listing. Print, rename, delete, browse, edit or view members 2 Data Set Allocate, rename, delete, catalog, uncatalog, or display information of en active data set. Print index listing. Print, rename, delete, catalog, uncatalog, or display	More:	+	
A Move/Copy Move, copy, or promote members or data sets A Dslist Print or display (to process) list of data set names. Print or display VTOC information			
5 Reset Reset statistics for members of ISPF library 6 Hardcopy Initiate hardcopy output 7 Transfer Download ISPF Client/Server or Transfer data set 8 Outlist Display, delete, or print held job output 9 Commands Create/change and application command table * Reserved This option reserved for future expansion 11 Format Format			
12 SuperC Compare data sets (Standard Dialog) 13 SuperCE Compare data sets (Extended Dialog) 14 Search-For Search data sets for strings of data (Standard Dialog) F1=Help F2=Split F3=Exit F7=Backward F8=Forward F9=Swap F10=Actions F12=Cancel F1 F1			
Recording macro	om:623	J	

図 6. 「ユーティリティー選択パネル (Utility Selection Panel)」アプリケーション画面

上記の図では、ISPF は、このメニューの「オプション (Option)」行にテキス ト・カーソルを自動的に置きます。画面のオプション行にテキスト・カーソル がありますか。ない場合、オプション行をクリックして、テキスト・カーソル をオプション行に移動させてください。

- 8. 4 と入力し、Enter キーを押す。上記の図に表示されているように、4 は Dslist を選択します。
- 9. アプリケーション画面が、「データ・セット・リスト・ユーティリティー (Data Set List Utility)」に変わります。 16 ページの図 7 を参照してくださ

い。

3270 Diplay - A - TN03001
<u>Fi</u> le <u>E</u> dit <u>V</u> iew <u>C</u> ommunication <u>A</u> ctions <u>H</u> elp
▣ ▣ ₽ # № ₽ ₩
ispf_ex1 🐱 🕹 💩 🕹 🖶 👹
<u>M</u> enu <u>R</u> efList R <u>e</u> fMode <u>U</u> tilities <u>H</u> elp
Option ===>
blankDisplay data set listPDisplay data set listvDisplay VTOC informationPVDisplay VTOC information
Enter one or both of the parameters below: Dsname Level V olume serial WORK01
Data set list options Initial View 2 3. Attrib 4. Total 1. Volume 2. Space 3. Attrib Enter " / " to select option /_ Confirm Data Set Delete Confirm Member Delete
When the data set list is displayed, enter either: " / " on the data set list command field for the command prompt pop-up, an ISPF line command, the name of a TSO command, CLIST, or REXX exec, or " = " to execute the previous command.
F1=Help F2=Split F3=Exit F7=Backward F8=Forward F9=Swap F10=Actions F12=Cancel
mvs059.raleigh.ibm.com:623

図 7. 「データ・セット・リスト・ユーティリティー (Data Set List Utility)」アプリケーション画面

- 「マクロの再生または記録を停止 (Stop playing or recording macro)」アイコンをクリックして、記録を停止します。これは、黒い四角形の下にカセットの画像が表示されているアイコンです。マクロ・マネージャー・ツールバーの右側の5つのアイコンが使用不可になり、左側の5つのアイコンが使用可能になります。
- 11. これで記録が完了しました。図 7 の「データ・セット・リスト・ユーティリティー (Data Set List Utility)」画面が表示されます。

観察結果:

- 記録を開始するために、「マクロを記録 (Record macro)」アイコンをクリック しました。
- アプリケーションを使用する場合の通常の操作とまったく同じように、入力しました (マウスのクリックとキー・ストローク)。ホスト・アプリケーション (ISPF) は、通常通りにアプリケーション画面を表示し、ユーザー入力に応答しました。
- マクロ・オブジェクトは、アプリケーション画面(もっと正確に言えば、各アプリケーション画面の複数の識別特性)を記録し、ユーザーが入力した通りにマウスのクリックとキー・ストロークを記録しました。
- マクロの記録を停止するために、「マクロの再生または記録を停止 (Stop playing or recording macro)」アイコンをクリックしました。

同じマクロに複数のセッションとの対話を記録できます。これは拡張機能です (191 ページの『複数のセッションとの対話』を参照)。

単純マクロの再生

ここでは、記録したマクロを再生する方法を示します。作業を開始する前に、 「ISPF 基本オプション・メニュー (ISPF Primary Option Menu)」に戻ってください。これが、このマクロの開始点です。

- アプリケーション・サーバーが ISPF 基本オプション・メニューであることを確認する。 14 ページの図 5 を参照。
- 実行するマクロを選択する。この例で使用したマクロを記録した場合、そのマクロの名前が、「現在選択されているマクロ (currently selected macro)」フィールド (マクロ・マネージャー・ツールバーの左側にある白いテキスト・フィールド) に表示されます。表示されない場合は、下記の手順を実行して、そのマクロを現在選択されているマクロにしてください。
 - マクロ・マネージャー・ツールバーで、「マクロを選択 (Select a Macro)」 をクリックする。このアイコンは、大きな下向き矢印です。
 - 「使用可能なマクロ」ウィンドウが表示されます。
 - 「マクロの場所」の下で、マクロの場所、例えばパーソナル・ライブラリー をクリックする。
 - 「マクロ・リスト (Macro List)」の下で、この章の直前のセクションで記録 されたマクロに割り当てた名前 (例えば、ispf_ex1.mac) をクリックする。
 - 「OK」をクリックする。
- 3. 実行したいマクロの名前が、現在選択されているマクロとして表示されているこ とを確認する。
- 「マクロを再生 (Play macro)」をアイコンをクリックして、選択されたマクロ を再生する。 (このアイコンは、カセットの画像の上に小さい右向き矢印が表示 されています。)
- アプリケーション画面が、すばやく「ユーティリティー選択パネル (Utility Selection Panel)」画面に変わり、次に「データ・セット・リスト・ユーティリ ティー (Data Set List Utility)」画面に変わります。また、再生中に、マクロ・ マネージャー・ツールバーの左側にあるアイコンが、一時的に使用不可になりま す。再生が完了した後、これらのアイコンは、再び使用可能になります。
- 6. これで再生が完了しました。

観察結果:

- マクロを再生する前に、特定のアプリケーション画面を表示する必要がありました。単純マクロではほとんど当てはまるように、マクロ再生の開始点は、マクロの記録を開始した点でもあります。結局、このユーザー入力が意味をなすのは、 所定のコンテキスト、つまり、マクロが記録されたコンテキストだけです。
- 次の方法でマクロを再生しました。
 - 「マクロを選択 (Select a macro)」アイコンをクリックしてから、特定のマ クロを選択する。
 - 「マクロを再生 (Play macro)」をクリックして、選択されたマクロを再生する。

- マクロの再生中に、マクロ・ランタイムは、以前に記録したマウス・クリックと キー・ストロークを再作成しました。アプリケーションは、通常どおりに応答し ました。アプリケーションは、人間による入力とマクロ・ランタイムによる入力 を再生時に区別できませんでした。
- マクロを再生することによって、ある ISPF メニューから、別のメニューを経由して、3 番目のメニューに迅速かつ正確に移動するアクションを実行できました。

キーの組み合わせへのマクロの割り当て

Host On-Demand では、特定のキー・ストロークの組み合わせにマクロを割り当て ることができます。記録したマクロをキー・ストロークの組み合わせに割り当てる 手順は、次のとおりです。

- 「編集 (Edit)」>「設定 (Preferences)」>「キーボード (Keyboard)」の順にクリ ックする。「キーボード (Keyboard)」ウィンドウが表示されます。
- 2. 「キー割り当て (Key Assignment)」タブをクリックする。
- 3. 「カテゴリー (Category)」リスト・ボックスで、Macros を選択する。
- 4. マクロのリストで、キーを割り当てたいマクロの名前 (例えば、ispf_ex1.mac) を選択する。
- 5. 「キーを割り当てる (Assign a Key)」をクリックする。「キーを押してください (Press a key)」というメッセージが表示されます。
- 6. Ctrl+i と入力する。このキー・シーケンスを入力した後、Ctrl+I というラベル がマクロ名の下に表示されます。
- 7. 「保管 (Save)」をクリックして、この割り当てを保管する。
- 8. 「OK」をクリックして、キーボード・ウィンドウを閉じる。

割り当てられたキーの組み合わせを使用してマクロを再生する手順は、次のとおり です。

- 1. アプリケーションをこのマクロの開始点 (「ISPF 基本オプション・メニュー (ISPF Primary Option Menu)」) に置く。
- 2. Ctrl+i を押す。
- 3. マクロが再生されます。

第4章 マクロの構造

この章には、2 つの目的があります。1 つは、マクロが XML マクロ・スクリプト 内で表示されるとおりにマクロの一般的な構造を説明することです。もう 1 つは、 マクロ・エディターと、マクロ・スクリプト内の特定の XML エレメント間の関連 付けの一部を示すことです。

- 『マクロ・スクリプト』では、<HAScript> エレメント、およびそのエレメント とマクロ・エディターの「マクロ (Macro)」タブとの関連付けについて説明しま す。
- 24 ページの『マクロ画面とそのサブコンポーネント』では、<screen> エレメント、およびそのエレメントとマクロ・エディターの「画面 (Screens)」タブとの関連付けについて説明します。

マクロ・スクリプト

マクロ・スクリプトは、Host On-Demand マクロの保管に使用される XML スク リプトです。マクロ・スクリプトの XML テキストを表示し、編集するには、コー ド・エディターを使用します (10 ページの『コード・エディター』を参照)。マク ロ・エディターは、コード・エディターに表示されるものと同じ情報を表示しま す。ただし、マクロ・エディターは、グラフィカル・ユーザー・インターフェース のリスト・ボックス、チェック・ボックス、入力フィールド、およびその他の制御 機構を使用して、コード・エディターよりも使いやすい形式で情報を表示します (9 ページの『マクロ・エディター』を参照)。

マクロ言語の XML エレメントについて少し学習すると、次のものを含めて、重要なトピックの理解がはるかに深まります。

- マクロ・エディターの使用方法
- マクロ再生の働き
- 有効なマクロの作成方法

したがって、本書では、マクロ・エディターの入力フィールド、ボタン、およびリ スト・ボックスだけでなく、同じ情報が保管されている対応する XML エレメント も頻繁に参照します。

XML エレメント

マクロ・スクリプトを理解するには、XML について多くを学習する必要はありません。構文の基本だけで十分です。 XML 構文の知識を復習する必要がある場合は、 209 ページの『Host On-Demand マクロ言語の XML 構文』で詳しく学習できます。しかし、必要な情報はほとんどすべて、このサブセクションに記載されています。

おそらくすでにご存じのように、XML スクリプトは、XML エレメントの集合から 構成されます。XML エレメントの一部には、他の XML エレメントが含まれてい ます。これは、一部の HTML エレメントに他の HTML エレメントが含まれてい るのとほぼ同じです。しかし、HTML の場合と異なり、XML では、プログラム開 発者が、保管したい情報の構造を表す新しい XML エレメントを定義できます。 Host On-Demand マクロ言語には、マクロの記述に必要な情報を保管するために、 約 35 種類の XML エレメントが含まれています。このマクロ言語については、 207 ページの『第 3 部 マクロ言語』の長さで詳しく説明しています。

XML マクロ・エレメントを参照する場合、本書では、エレメント名を不等号括弧で 囲んで使用します。例えば、<HAScript> エレメント、<screen> エレメントなどで す。

図 8 は、XML エレメントの例を示しています。

<SampleElement attribute1="value1" attribute2="value2">

</SampleElement>

図 8. サンプル XML エレメント

上記の図に表示されている <SampleElement> エレメントには、すべてのマクロ・ エレメントのキー・コンポーネントが含まれています。最初の行は、開始タグで す。開始タグは、左不等号括弧 (<) の後に、XML エレメントの名前 (SampleElement)、属性定義、右不等号括弧 (>) の順に続きます。 2 行目の省略符 号 (...) は、XML 構文の一部ではなく、上記の図では、<SampleElement> エレメン ト内に他のエレメントが存在する可能性があることを示すために使用されます。 3 行目は、終了タグです。終了タグでは、エレメント名が不等号括弧で囲まれ、最初 の不等号括弧の後にスラッシュが付きます (</Sample Element>)。

開始タグでは、属性を指定するのに、属性名 (例えば、attribute1)の後に、等号 (=)、引用符で囲まれた属性値 (例えば、"value1")の順に続きます。開始タグでは、 任意の数の属性を指定できます。

マクロ・エレメントに他の XML エレメントが含まれていない場合、図 9 のよう に、省略形式で書き込むことができます。

<SampleElement attribute1="value1" attribute2="value2" />

図 9. 省略形式で書かれたサンプル XML エレメント

上記の図では、<SampleElement> エレメントは、左不等号括弧 (<) の後に、名前 (SampleElement)、属性、スラッシュ、右不等号括弧 (/>) の順に続けて書き込まれ ています。このように、XML エレメント全体が 1 対の不等号括弧内に書き込まれ ます。

マクロ・スクリプトの概念視点

マクロ・スクリプトは、最高 3 つのタイプのサブエレメントを含む 1 つの <HAScript> エレメントから構成されます。

- 1 つの <import> エレメント (オプショナル)
- 1 つの <vars> エレメント (オプショナル)
- 1 つ以上の <screen> エレメント

図 10 は、3 つの <screen> エレメントを含むサンプル・マクロ・スクリプトの概 念視点を示しています。

HAScript	
	Import
	変数
	Screen1
	Screen2
	Screen3

図 10. マクロ・スクリプトの概念視点

上記の図は、主なタイプのサブエレメントのインスタンスが含まれている <HAScript> エレメント (HAScript) を示しています。これらのサブエレメントは、 <import> エレメント (Import)、<vars> エレメント (Variables)、および 3 つの <screen> エレメント (Screen1、Screen2、および Screen3) です。

すべてのマクロ・スクリプトは、上記のような構造を持っています。ただし、大部 分のマクロ・スクリプトには、もっと多くの screen があります。上記のマクロで 50 個の screen がある場合、上記の図の外観はほぼ同じですが、Screen3 の後に、 追加の screen (Screen4、Screen5 から、Screen50 まで)が続きます。 (ただし、 screen が保管される順序は、必ずしも、マクロの再生時に screen が実行される順 序を表しているわけではありません。)

<HAScript> エレメントは、マクロ・スクリプトのマスター・エレメントです (HAScript は、Host Access Script を意味します)。このエレメントは、マクロ全体 を囲み、その開始タグには、マクロ全体に適用できる情報 (例えば、マクロの名前) を含む属性が入っています。 <HAScript> エレメントの例については、 24 ページ の図 12 を参照してください。

<import> エレメントは、Java クラスのインポートに使用され、オプションです。 Java クラスのインポートは、上級トピックであり、 158 ページの『Java クラスの インポート型の作成』で説明します。

<vars> エレメントは、標準データ型 (boolean、integer、double、string、または field) のいずれかに属する変数を宣言し、初期化するのに使用されます。標準変数 の使用は、上級トピックであり、 153 ページの『第 11 章 変数とインポートした Java クラス』で説明します。

<screen> エレメントは、マクロ画面の定義に使用されます。 <screen> エレメント は、<HAScript> 内にある最も重要なエレメントです。上記の図 10 で分かるよう に、マクロ・スクリプトは、主に <screen> エレメント (例えば、図中の Screen1、Screen2、および Screen3) から構成されます。また、マクロ・スクリプト 内のその他の種類の XML エレメントの大部分も、<screen> エレメント内で指定さ れます。

マクロ・タブの概要

マクロ・エディターの操作に慣れるために、この節では、マクロ・エディターの 「マクロ (Macro)」タブと、前の節で説明されている <HAScript> エレメントとの 非常に単純な比較で成り立っています。

マクロ・エディターには、4 つのタブがあります。「マクロ (Macro)」、「画面 (Screens)」、「リンク (Links)」、および「変数 (Variables)」です。最初のマク ロ・タブは、<HAScript> エレメントと非常に密接に対応します。実際に、マク ロ・タブは、<HAScript> エレメントの開始タグの属性に保管される情報用のグラ フィカル・ユーザー・インターフェースです。

したがって、<HAScript> エレメントはマクロ・スクリプトのマスター・エレメン トであり、マクロ全体に適用される情報 (例えば、マクロ名) がその属性に含まれる ので、同様に、マクロ・タブは、マクロ・エディターの最初のタブであり、同じグ ローバル情報の一部にアクセスできます。

23 ページの図 11 は、マクロ・タブが選択された状態のマクロ・エディターを示しています。
Macro	Screens	Links	Variables	
No. Stor	1602.000	10.00		
			Macro Na	me fang avl
			WIGCTO ING	
			Descriptio	n
			Author	Mahesh
			Creation [Date 14, 2015 1:03:32 AM
			V Paus	e Between Actions 300 milliseconds
			J Time	out Between Screens 60000 millineconds
				Nu between screens
			Show	v All Prompts at Start of Macro
			Use \	Variables and Arithmetic Expressions In Macro
	Save and F	xit	Save As Sa	ve Cancel Code Editor Import Export Help



上記の図では、マクロ・タブには、「マクロ名 (Macro Name)」、「記述 (Description)」、およびその他の情報の入力フィールド、および複数のチェック・ ボックスがあります。次の 2 つのフィールドに注目してください。

- 「マクロ名 (Macro Name)」フィールドには、マクロに割り当てる名前が入ります。この名前は、マクロを編集または実行するときに選択する名前と同じです。 マクロ名は大/小文字が区別されます。例えば、ispf_usp は Ispf_usp、 ispf_usP、ISPF_USP などとは異なる名前です。
- 「マクロで変数および演算式を使用」チェック・ボックスは、マクロ・オブジェ クトがこのマクロに基本マクロ形式を使用するか、拡張マクロ形式を使用するか を決定します。上記の図では、このチェック・ボックスは選択されていません。 これは、基本マクロ形式が使用されることを示します(33ページの『マクロ形 式の選択』を参照)。

24 ページの図 12 は、図 11 のマクロ・タブに表示されているのと同じ情報およびいくつかの追加情報が含まれている、サンプル <HAScript> エレメントを示しています。コード・エディターでは、<HAScript> エレメントは 1 行に書き込まれま

すが、ここでは、属性が見えるように複数行で書かれています。

```
<HAScript

name="ispf_ex1"

description=" "

timeout="60000"

pausetime="300"

promptall="true"

author=""

creationdate=""

supressclearevents="false"

usevars="false"

ignorepauseforenhancedtn="false"

delayifnotenhancedtn="0">
```

</HAScript>

図 12. サンプル <HAScript> エレメント

上記の図の <HAScript> エレメントには、 23 ページの図 11 に表示されているマ クロ・タブの各入力フィールドに対応する属性があることに注目してください。例 えば、<HAScript> エレメント内の name 属性 (name="ispf_ex1") は、マクロ・タ ブのマクロ名フィールドに対応します。同様に、<HAScript> エレメント内の usevars 属性 (usevars="false") は、マクロ・タブの「変数と演算式を使用する」 チェック・ボックスに対応します。

マクロ画面とそのサブコンポーネント

ここでは、マクロ画面とその主なサブコンポーネントについて説明します。マクロ 画面の定義は、定義が必要な別の用語、すなわちアプリケーション画面によって決 まります。

アプリケーション画面

アプリケーション画面は、ホスト・アプリケーションによって Host On-Demand セッション・ウィンドウ上に表示される、意味のある文字配置です。

おそらくお気付きのように、アプリケーション画面の概念はすでに十分理解してい ます。アプリケーション画面の一例は、 25 ページの図 13 に表示されている 「ISPF 基本オプション・メニュー (ISPF Primary Option Menu)」です。 (これと 同じアプリケーション画面は、 14 ページの図 5 に表示されています。)

西 등 대 종 학 현 등 湖 ipg_et1 Op t	■ * * * * * * * * ▼ * * * * * * *	⊕ ≝≝∑∯∎ s <u>C</u> ompilers <u>O</u> ptions <u>S</u> tatus <u>H</u> elp	
ispg_ex1	▼ ∡¥ ≱ ≩ ₫	≝≝∑o∰ s <u>C</u> ompilers <u>O</u> ptions <u>S</u> tatus <u>H</u> elp	
 Opt	enu <u>U</u> tilitie	s <u>C</u> ompilers <u>O</u> ptions <u>S</u> tatus <u>H</u> elp	
 Opt	enu <u>U</u> tilitie	s <u>C</u> ompilers <u>O</u> ptions <u>S</u> tatus <u>H</u> elp	
Opt	enu <u>U</u> tilitie	s <u>C</u> ompilers <u>O</u> ptions <u>S</u> tatus <u>H</u> elp	
Opt			
Opt		OS/390 Primary Option Menu	
	ion ===>	Soroso in filling option here	
		More:	+
0	Settings	Terminal and User parameters	User ID . : USER1
1	View	Display source data or listings	Time : 14:31
2	Edit	Create or change source data	Company 1
3	Utilities Foreground	Interactive language processing	Screen : I
7	Ratch	Submit job for language processing	Appl ID · ISP
5	Command	Enter TSO or Workstation commands	$\frac{1}{1000} = \frac{1}{1000} = \frac{1}{1000} = \frac{1}{1000}$
7	Dialog Test	Perform dialog testing	TSO prefix: USEB1
8	LM Facilitu	Library administrator functions	System ID : MVS011
9	IBM Products	IBM program development products	MVS acct. : MVSBLD
10	SCLM	SW Configuration Library Manager	Release . : ISPF 7.1
LI	censed Materi	als - Property of IBM atio	ns
00	SU-ZUS LU	pyright IBM Corp. 1980, 2013.	
03	e dunlicatio	n or disclosure restricted	
bu	GSA ADP Sche	dule Contract with IBM Corp. 7.2	
		ard	F8=Forward F9=Swap
F10	=Actions F12		
MA	a		18/032

図 13. サンプル・アプリケーション画面「ISPF 基本オプション・メニュー (ISPF Primary Option Menu)」

上記の図では、このアプリケーション画面には、一番上の行にメニュー選択項目 (Menu、Utilities、Compilers、Options など)、一番下の行にファンクション・キー の割り当て (F1=Help、F2=Split など)、上部付近にタイトル (ISPF Primary Option Menu)、左側にオプションのリスト (0 から V)、およびオプション番号または文字を 入力する入力フィールド (Option ===>) が表示されています。ユーザーが入力する と (例えば、3 (Utilities を表す) を入力した後、Enter キーを入力すると)、ISPF ア プリケーションは、セッション・ウィンドウからこれらの表示項目をすべて除去 し、別のアプリケーション画面を表示します。

マクロ画面

マクロ画面は、特定のアプリケーション画面へのアクセスを管理する方法をマク ロ・ランタイムに指示する、1 組の命令です。マクロ画面には、次のものが含まれ ています。

- 特定のアプリケーション画面の記述
- この特定のアプリケーション画面にアクセスするときに取ることができるアクション
- この特定のアプリケーション画面の後に表示されるマクロ画面のリスト

この時点では、概念はあまり直観的ではありませんが、同じアプリケーション画面 を参照する複数のマクロ画面が同じマクロに存在する場合があります。マクロ画面 が相互にリンクされる方法によって、マクロ・ランタイムは、マクロの再生時に、 同じアプリケーション画面に複数回アクセスして、アクセスごとに異なるマクロ画 面を処理する場合があります。

また、1 つのマクロ画面が、複数のアプリケーション画面を参照する場合もありま す。複数のアプリケーション画面が互いに類似している場合、マクロ開発者は、類 似したすべてのアプリケーション画面を処理するマクロ画面を作成することができ ます。 それにもかかわらず、各マクロ画面は、なんらかのアプリケーション画面に対応し ます。マクロを記録する際には、マクロ・オブジェクトは、記録中に、ユーザーが アクセスするアプリケーション画面ごとにマクロ画面を作成し、保管します。同じ アプリケーション画面に複数回アクセスする場合、マクロ・オブジェクトは、アク セスごとにマクロ画面を作成し、保管します。同様に、記録されたマクロを再生す る際には、マクロ・ランタイムは、再生中に、アクセスするアプリケーション画面 ごとに1 つのマクロ画面を処理します。

マクロ画面の概念視点

マクロ画面は、次の 3 つの必須サブエレメントを含む 1 つの <screen> エレメントから構成されます。

- 1 つの <description> エレメント (必須)
- 1 つの <actions> エレメント (必須)
- 1 つの <nextscreens> エレメント (必須。ただし、出口画面を除く)

これらのサブエレメントは必須であり、それぞれ 1 つしか指定できません。

図 14 は、<screen> エレメントの概念視点を示しています。

Sci	reen1
	記述
	アクション
	有効な次画面

上記の図は、<screen> エレメント (Screen1) に 3 つの必須サブエレメント (<description> エレメント (記述)、<actions> エレメント (アクション)、および <nextscreens> エレメント (有効な次画面) が含まれていることを示しています。

すべての <screen> エレメントは、これらの 3 つのサブエレメントを使用して上記 のような構造を持っています。 (4 番目のオプションのサブエレメント <recolimit> エレメントについては、本書のこれ以降の部分で説明します。)

<screen> エレメントは、マクロ画面のマスター・エレメントです。このエレメント には、その特定のマクロ画面に属している他のすべてのエレメントが含まれていま す。また、開始タグには、マクロ画面全体に適用できる情報 (例えば、マクロ画面 の名前) が入っている属性も含まれています。

<description> エレメントには、<description> エレメントが属している <screen> エレメントが、特定のアプリケーション画面に関連していることを、マクロ・ラン タイムが認識できるようにするディスクリプターが含まれています。このディスク リプターと <description> エレメントについては、 53 ページの『第 7 章 画面記 述と画面認識』で説明しています。

<actions> エレメントには、マクロ・ランタイムがアプリケーション画面で実行す る各種アクション (例えば、アプリケーション画面からのデータの読み取りや、キ

図 14. <screen> エレメントの概念視点

ー・ストロークの入力) が含まれています。これらのアクションと <actions> エレ メントについては、 77 ページの『第 8 章 マクロ・アクション』で説明していま す。

<nextscreens> エレメント (26 ページの図 14 内の Valid Next Screens) には、 現行のマクロ画面の後で表示される可能性があるすべての <screen> エレメントの 画面名のリストが含まれています。 <nextscreens> エレメントと、そのエレメント の中に入っているエレメントについては、 137 ページの『第 9 章 画面認識、パー ト 2』で説明しています。

画面タブの概要

このセクションでは、マクロ・エディターの「画面 (Screens)」タブと、直前のセク ションで説明されている XML <screen> エレメントとの関連を示します。 図 15 は、画面タブが選択された状態のマクロ・エディターを示しています。

ホスト・アクセス・マクロ・エディター	×
マクロ	
画面名 Screen1 画面を削除	
一般 記述 アクション	
画面名 Screen1	
入り口画面 true 💌 出口画面 false 💌	
一時画面 false v	
認識限界を設定する エラーまでの画面数	
└──休止時間を設定する └───────── 休止時間 (ミリ秒)	
保管して終了 別名保管 保管 キャンセル コード・エディター インポート エクスポート ヘルプ	
マクロに組み込まれる画面を定義する	

図 15. マクロ・エディターの画面タブ

上記の図では、画面タブに次の項目が入っています。

- このタブの上部にある「画面名 (Screen Name)」リスト・ボックス
- 3 つの従属タブ (「一般 (General)」、「記述 (Descriptions)」、および「アクション (Actions)」)

現在、一般タブが選択されています。

画面タブには、次の 2 つの「画面名 (Screen Name)」フィールドがあることに注目 してください。

- ・ 画面タブの一番上にある「画面名 (Screen Name)」フィールドは、マクロ内のす べてのマクロ画面の名前が入っているリスト・ボックスです。
- 「一般 (General)」サブタブの一番上にある「画面名 (Screen Name)」フィール ドは、現在選択されている画面に割り当てたい名前を入力する入力フィールドで す。

画面タブの一番上にある画面名リスト・ボックスでは、使用したいマクロ画面の名 前 (例えば、Screen1) をクリックします。マクロ・エディターは、そのマクロ画面 に属する情報をサブタブに表示します。例えば、 27 ページの図 15 では、リス ト・ボックスは、マクロ画面名 Screen1 を表示し、サブタブは、Screen1 に属する 情報を表示します。ユーザーがリスト・ボックス内の別のマクロ画面名 (おそら く、Screen10) を選択した場合、マクロ・エディターは、マクロ画面 Screen10 に属 する情報をサブタブに表示します。

一般タブの下の画面名入力フィールドには、現在選択されているマクロ画面に割り 当てたい名前を入力します。 Screenx のような画面名 (ここで、x は整数を表しま す。例えば、Screen1) は、一時的な名前であり、マクロ・オブジェクトがマクロ画 面を作成するときにその画面に指定する名前です。この名前をそのまま使用するこ とも、もっと覚えやすい記述名に置き換えることもできます。 (すべてのマクロ画 面に、Screen3、Screen10、Screen24 のような名前がある場合、どのマクロ画面が何 を実行するかを覚えるのは困難です。)

画面タブ上のサブタブ (「一般 (General)」、「記述 (Description)」、および「ア クション (Actions)」) は、直前のセクションで説明されている XML <screen> エ レメントの主要部分に対応します。具体的には、次のとおりです。

- 「一般 (General)」サブタブは、<screen> エレメントの属性に保管されている情報を提示します。
- 「記述 (Description)」サブタブは、<screen> エレメントの <description> サブ エレメントに保管されている情報を提示します。
- 「アクション (Actions)」サブタブは、<screen> エレメントの <actions> サブエ レメントに保管されている情報を提示します。

しかし、<nextscreens> サブエレメントはどうでしょうか。使いやすくするため に、<nextscreens> エレメントに属する情報は、上位の「リンク (Links)」タブに表 示されます。 27 ページの図 15 の画面タブのすぐ右に、リンク・タブが表示され ています。

図 16 は、Screen1 という名前のサンプル <screen> エレメントの XML 開始タグ と終了タグを示しています。

<screen name="Screen1" entryscreen="true" exitscreen="false" transient="false">
...
</screen>

図 16. <screen> エレメントの開始タグと終了タグ

上記の図では、省略符号 (...) は、XML テキストの一部ではなく、簡単にするため に、<screen> エレメント内に含まれている必須エレメントが省略されていることを 示しています。開始タグ内の属性は、 27 ページの図 15 の一般タブ上のフィール ドに対応していることに注意してください。例えば、name 属性 (name="Screen1") は、一般タブ上の画面名入力フィールドに対応し、entryscreen 属性 (entryscreen="true") は、一般タブ上の「入り口画面 (Entry Screen)」リスト・ボ ックスに対応します。

図 17 は、囲まれているエレメントを含めて、<screen> エレメント全体の XML テ キストを示しています。

図 17. サンプル XML <screen> エレメント

上記の図では、<screen> エレメントに、必須の <description>、<actions>、および <nextscreens> エレメントが入っていることに注目してください。

第2部マクロの開発

第5章 データ型、演算子、および式

マクロ形式の選択

基本マクロ形式と拡張マクロ形式の比較

マクロを保管する形式 (基本マクロ形式または拡張マクロ形式) を選択する必要があります。

デフォルトの形式は基本マクロ形式です。基本マクロ形式は、基本的なレベルの機能をサポートしますが、拡張マクロ形式がサポートする式評価、変数、またはその他の機能をサポートしません。それにもかかわらず、式と変数を使用することがすでに分かっている場合を除いて、最初は基本マクロ形式を選択してください。後で簡単にマクロを拡張マクロ形式に切り替えることができます。(一方、拡張マクロ形式から始めた場合は、基本マクロ形式にマクロを切り替えるのがはるかに難しくなります。)

どちらの形式を使用するかを指定するには、マクロ・エディターのマクロ・タブ上 の「マクロで変数および演算式を使用」チェック・ボックスを使用します。

- 基本マクロ形式を選択するには、このチェック・ボックスのチェックマークを外す(デフォルト)。
- 拡張マクロ形式を選択するには、このチェック・ボックスにチェックマークを付ける。

基本マクロ形式では、次のことが可能です。

• 整数、倍精度、ブール (true または false)、およびストリングを含めて、リテラ ル値を入力できる。

一方、拡張マクロ形式では、次のことが可能です。

- 同様に、整数、倍精度、ブール (true または false)、およびストリングを含めて、リテラル値を入力できる。
- 「+」ストリング演算子を使用してストリング連結を可能にする
- 演算式を許可する
- 条件式を許可する
- 変数を許可する
- インポートされた Java 変数タイプとメソッドを許可する

ストリングと特殊文字の表記、演算子文字の取り扱い

基本マクロ形式を選択したか、拡張マクロ形式を選択したかに応じて、マクロ内で ストリングと 2 つの特殊文字 (単一引用符 (') と円記号 (¥))の書き込み方法が異な ります。また、基本マクロ形式では通常の文字である一部の文字は、拡張マクロ形 式では演算子として使用されます。 しかし、これらの規則が影響を与えるのは、次のタブにある入力フィールドだけで す。

- 「画面 (Screens)」タブの「記述 (Description)」タブ
- 「画面 (Screens)」タブの「アクション (Actions)」タブ
- 「変数 (Variables)」タブ

これらのタブ上で影響を受ける入力フィールドは次のとおりです。

- タブ上のテキスト入力フィールド (例えば、記述タブの「Field Counts and OIA」ウィンドウの「フィールド数 (Number of Fields)」テキスト入力フィールド)
- タブ上のリスト・ボックスで <Expression> 項目 (例えば、String ディスクリプ ター・ウィンドウ上の「大/小文字を区別しない (Ignore Case)」リスト・ボック ス内の <Expression> 項目) を選択するときに表示される、ポップアップ・ウィ ンドウ内のテキスト入力フィールド

上記にリストされているもの以外のすべてのタブ上の入力フィールドの場合は、常 に基本マクロ形式の規則を使用してください。

以下の 2 つのセクションでは、こうした規則の違いについて説明します。

基本マクロ形式のストリング表記規則

基本マクロ形式を選択した場合、記述タブ、アクション・タブ、および変数タブ上 の入力フィールドには、次の規則を使用します。

ストリングは、単一引用符で囲まずに書き込まれなければならない。例えば、次のとおりです。

apple banana To be or not to be John Smith

• 単一引用符 (') と円記号 (¥) は、先行する円記号なしに、その文字自体によって 表される。例えば、次のとおりです。

New Year's Day c:¥Documents and Settings¥User

次の文字または文字シーケンスは、演算子として扱われない。 +、-、*、/、%、
 ==、!=、>、<、>=、<=、&&、||、!

拡張マクロ形式のストリング表記規則

拡張マクロ形式を選択した場合、記述タブ、アクション・タブ、および変数タブ上 の入力フィールドには、次の規則を使用します。

すべてのストリングは、単一引用符で囲まれなければならない。例えば、次のとおりです。

```
'apple'
'banana'
'To be or not to be'
'John Smith'
```

単一引用符(')と円記号(¥)は、その文字自体に円記号を先行させて表記される。例えば、次のとおりです。

'New Year¥'s Day'
c:¥¥Documents and Settings¥¥User

- 次の文字または文字シーケンスは、演算子として扱われる。
 - ストリング連結演算子: +
 - 算術演算子: +、-、*、/、%
 - 条件演算子: ==、!=、>、<、>=、<=
 - 論理演算子: &&、||、!

別の形式へのマクロの変換

拡張マクロ形式へのマクロの変換

基本マクロ形式から拡張マクロ形式にマクロを変換するには、マクロ・タブの「マ クロで変数および演算式を使用」チェック・ボックスにチェックマークを付けるだ けです。その結果、マクロ・オブジェクトは次のことを行います。

- マクロのすべての拡張機能を使用可能にする。
- 変換が必要なすべての入力フィールドで、マクロ内のすべてのストリング、および2つの特殊文字(単一引用符(')と円記号(¥))のすべてのオカレンスを、基本表記から拡張表記に自動的に変換する。

つまり、マクロ・オブジェクトは、マクロ内のすべてのストリングを検出し、それ らを単一引用符で囲み、存在するすべての ' と ¥ を、¥' と ¥¥ に変更します。ま た、任意の演算子文字は演算子として扱われます。

基本マクロ形式へのマクロの変換

拡張マクロ形式から基本マクロ形式へのマクロの変換は、非常に困難な場合があり ます。「マクロで変数および演算式を使用」チェック・ボックスのチェックマーク を外しても、自動変換は行われません。自動的に行われるのは、次のことだけで す。

• マクロ・オブジェクトが、マクロのすべての拡張機能を使用不可にする。

ストリングおよび 2 つの特殊文字の表記はすべて、手作業で 1 つずつ基本表記に 戻す必要があります。また、マクロで拡張機能が使用されたインスタンスを削除す ることも必要です。削除しないと、スクリプトを保管または実行しようとするとき に、エラーまたは予期しない結果を検出する場合があります。残りの演算子文字は すべて、演算子としてではなく、リテラル文字として扱われます。

標準データ型

マクロ・オブジェクトは、次の標準データ型をサポートします。

- ブール (boolean)
- 整数 (integer)
- 倍精度 (double)
- ストリング (string)

以下のサブセクションで、これらのデータ型について説明します。

ブール・データ

ブール値 true と false は、大文字と小文字の任意の組み合わせで書き込むことが できます (例えば、True、TRUE、FALSE、falsE など)。

ブール値を必要とする入力フィールドの例は、画面タブの一般タブ上の入り口画面 フィールドです。条件を true に設定するには true を入力し、false に設定するに は false を入力してください。

ブール値はストリングでない

ブール値はストリングでないので、単一引用符で囲む必要はありません。基本マクロ形式を使用するか、拡張マクロ形式を使用するかに関係なく、ブールは常に、 true と false として書き込まれます。'true' と 'false' ではありません。

しかし、ブール・コンテキストでは、ストリング値はブール値に変換されます (40 ページの『ブールへの変換』を参照)。したがって、拡張マクロ形式では、ブー ル・フィールドにストリング 'true' を入力することができます。これは、マク ロ・エディターがストリング 'true' をブール値 true に変換するからです。

整数

整数は、コンマまたはその他の区切り文字なしに書き込まれます。例えば、次のと おりです。

10000

0 -140

整数定数

マクロ・エディターには、すべて大文字を使用して書き込まれる複数の整数定数が あります。これらの値は、ストリングとしてではなく、整数として扱われます。例 えば、次のとおりです。

- NOTINHIBITED
- FIELD_PLANE
- COLOR_PLANE

倍精度

倍精度は、コンマまたはその他の区切り文字なしに書き込まれます。例えば、次の とおりです。

ストリング

ストリングは任意の一連の文字であり、ブランク文字が先行したり、末尾に付いた り、間に入れたりすることができます。マクロが使用するように設定されている形 式が基本マクロ形式か、拡張マクロ形式かに応じて、一部の入力フィールド内のス トリングの表記が異なります。 33 ページの『ストリングと特殊文字の表記、演算 子文字の取り扱い』を参照してください。

次の例は、拡張マクロ形式の表記を使用しています。

^{3.1416} 4.557e5 -119.0431

```
'apple'
'User4'
'Total number of users'
' This string has 3 leading blanks.'
'This string has 3 trailing blanks. '
```

次は、基本マクロ形式の表記を使用した場合の同じ例です。

apple User4 Total number of users This string has 3 leading blanks. This string has 3 trailing blanks.

基本マクロ形式では、末尾ブランクを使用できますが、検出が困難であることに注 意してください。末尾ブランクがあるかどうか分からない場合は、コード・エディ ターでストリングの表記を参照してください。

フィールド

160 ページの『フィールド変数』を参照してください。

值 null

値 null は予約語であり、ストリングではありません。インポートされた Java ク ラスに属するオブジェクトの代わりに使用される場合、Java 言語における意味と同 じです。

空ストリングを表すのに、null を使用しないでください。空ストリングを表すに は、拡張マクロ形式では 1 対の単一引用符 ('') を使用し、基本マクロ形式では何も 使用しないでください。 (例えば、ストリング変数に割り当てることによって) スト リング・コンテキストで値 null を使用する場合、マクロ・エディターまたはマク ロ・ランタイムは、値 null をストリング 'null' に変換します。

算術演算子および式

演算式を使用するには、まず、マクロ・タブの「マクロで変数および演算式を使用」チェック・ボックスにチェックマークを付ける必要があります(33 ページの 『ストリングと特殊文字の表記、演算子文字の取り扱い』を参照)。

演算子および式

算術演算子は次のとおりです。

表	3.	算術演算子
		21114 12 12 14

演算子	働き
+	加算
-	減算
*	乗算
/	割り算
°%	モジュロ

演算式では、条件は左から右に評価されます。演算子の優先順位は *、/、%、+、-です。例えば、次の演算があるとします。

4 * 2 + 16 / 8 - 1 * 2

この結果は 8 です。次のように小括弧を使用すると、式が評価される順序を指定で きます。

(4 * 2) + (16 / 8) - (1 * 2) evaluates to 8 4 * ((2 + 16) / (8 - 1)) * 2 evaluates to 20.571

演算式の使用場所

演算式は、算術値を使用できる場所であれば、ほとんどどこでも使用できます。例 えば、次のとおりです。

- 画面のパラメーターとして。例えば、次のとおりです。
 - 画面の認識限界を指定する。
 - 画面の休止時間を指定する。
- ディスクリプターのパラメーターとして。例えば、次のとおりです。
 - カーソル・ディスクリプターの行または列を指定する。
 - 数値フィールド・カウント・ディスクリプター内のフィールド数を指定する。
 - 属性ディスクリプターの行、列、または属性値を指定する。
 - 条件ディスクリプターの条件として。
- アクションのパラメーターとして。例えば、次のとおりです。
 - マウス・クリック・アクションで行または列を指定する。
 - ボックス選択アクションで行または列を指定する。
 - 休止アクションのミリ秒数として。
 - 変数更新アクションで値を指定する。
 - 条件アクションの条件として。
- 変数の初期値として。

ストリング連結演算子 (+)

ストリング連結演算子「+」を使用できるのは、マクロ・タブの「マクロで変数およ び演算式を使用」チェック・ボックスにチェックマークを付ける場合だけです。 33 ページの『基本マクロ形式と拡張マクロ形式の比較』を参照してください。

演算子および式

次の表は、ストリング演算子を示しています。

表 4. 算術演算子

演算子	働き
+	連結

複数の連結が入っているストリング式を作成することができます。次の例では、拡 張形式に必要なストリング表記を使用します (33 ページの『ストリングと特殊文 字の表記、演算子文字の取り扱い』を参照)。 Expression:

Evaluates to:

'Hello ' + 'Fred' + '!' 'Hello Fred!' 'Hi' 'There' (Error, a + operator is required to concatenate strings) 'Hi' + 'There' 'HiThere'

条件演算子と論理演算子および式

条件演算子は次のとおりです。

表 5. 条件演算子

演算子	働き
==	等しい
!=	等しくない
>	より大
<	より小
>=	より大か等しい
<=	より小か等しい

論理演算子は次のとおりです。

表 6. 論理演算子

演算子	働き
&&	AND
11	OR
!	NOT

HTML または XML エディターで && と入力する場合は、&& と入力 する必要がある場合があります。

条件式では、条件は左から右に評価されます。演算子の優先順位は、上記の表にリ ストされている順序と同じです。次のように小括弧を使用すると、式が評価される 順序を指定できます。例えば、次のとおりです。

Expression: Evaluates to:

(4 > 3) true !(4 > 3) false (4 > 3) && (8 > 10) false (4 > 3) || (8 > 10) true

条件式には複合条件を含むことができる

条件式には、演算式、変数、およびインポートされた Java クラスのメソッドの呼び出しを含むことができます。

条件式の用途

条件演算子と論理演算子は、次の 2 つのコンテキストでしか使用できません。

- 条件ディスクリプターの条件フィールド
- 条件アクションの条件フィールド

自動データ型変換

コンテキストの影響

データの項目が1つの標準データ型(ブール、整数、倍精度、またはストリング) に属しているにもかかわらず、コンテキストが別の標準データ型を必要とする場 合、データの評価時(マクロ・エディターがデータを保管するとき、またはマク ロ・ランタイムがマクロを再生するとき)に、可能な場合は、コンテキストが必要 とする標準データ型に自動的に変換されます。

コンテキストの例は次のとおりです。

- 条件ディスクリプターの条件フィールド (ブール値を期待する)
- メッセージ・アクションのメッセージ・テキスト・フィールド (ストリング値を 期待する)
- 変数がフィールド変数である場合、変数更新アクションの値フィールド (ロケーション・ストリングを期待する)
- 入力アクションの行値 (整数値を期待する)

しかし、データを新しいデータ型に変換できない (例えば、ストリング 123apple を整数に変換できない) 場合は、エラーが発生します。マクロ・エディターがエラ ー・メッセージを表示します。マクロ・ランタイムは、マクロの再生を停止し、エ ラー・メッセージを表示します。

次のサブセクションでは、標準データ型ごとの変換について説明します。

ブールへの変換

ブール・コンテキスト内のストリング 'true' (または 'TRUE'、'True' など) は、 ブール true に変換されます。ブール・コンテキスト内のその他のすべてのストリ ング ('false'、'1'、'apple' など) は、ブール false に変換されます。

'true' (in an input field that requires a boolean) converts to true 'apple' (in an input field that requires a boolean) converts to false

整数への変換

整数コンテキスト内の有効な整数形式のストリングは、整数に変換されます。

' 4096 '	converts to	4096
'-9'	converts to	-9

倍精度への変換

倍精度コンテキスト内の有効な倍精度形式のストリングは、倍精度に変換されま す。

'148.3' converts to 148.3

倍精度と結合された整数は、倍精度になります。

10 + 6.4 evaluates to 16.4

ストリングへの変換

ストリング・コンテキスト内のブール、整数、または倍精度は、ストリングに変換 されます。 (ブール値 true および false は、ストリングでないことに注意してくだ さい。 36 ページの『ブール・データ』を参照してください。)

'The result is ' + true evaluates to 'The result is true'
FALSE (in an input field that requires a string) converts to 'FALSE'
'The answer is ' + 15 evaluates to 'The answer is 15'
22 (in an input field that requires a string) converts to '22'
('4.5' == .45e1) evaluates to true
14,52 (in an input field that requires a string) evaluates to'14,52'

変換エラー

コンテキストに変換が必要であるにもかかわらず、データの形式がその変換に有効 でない場合、マクロ・エディターはエラー・メッセージを表示します。マクロの再 生中にエラーが発生すると、マクロ・ランタイムは、エラー・メッセージを表示 し、そのマクロを終了させてランタイム・エラーを出します。

'123apple'	in an integer context	Error
'22.7peach'	in a double context	Error

等価

特定の標準データ型の即時値を受け入れるコンテキストはいずれも、同じデータ型 の任意のエンティティーも受け入れます。

例えば、入力フィールドがストリング値 (例えば、'Standard Dialog') を受け入れ る場合、次のものも受け入れます。

- ストリングに評価される式
- ストリングに変換される値
- ストリング変数
- ストリングを戻すインポート・メソッドの呼び出し

同様に、入力フィールドがブール値 (true または false) を受け入れる場合、次のものも受け入れます。

- ブール値に評価される式
- ブール値に変換される値
- ブール変数
- ブールを戻すインポート・メソッドの呼び出し

マクロ機能におけるこうした柔軟性を認識すると、さらに強力なマクロを作成する のに役立ちます。

行または列の負の値の意味

ストリング・ディスクリプターやその他の複数のディスクリプターやアクションで は、セッション・ウィンドウの行または列の負の値は、セッション・ウィンドウの 最後の行または最後の列からのオフセットを示します。マクロ・ランタイムは、行 または列の位置を次のように計算します。 actual row = (number of rows in text area) + 1 + (negative row offset)
actual column = (number of columns in text area) + 1 + (negative column offset)

例えば、セッション・ウィンドウに 24 行のテキストがある場合、行座標 -1 は、 実際の行座標 24 を示します (24 + 1 - 1 として計算)。同様に、セッション・ウィ ンドウに 80 列のテキストがある場合、列座標 -1 は、実際の列座標 80 を示しま す (80 + 1 - 1 として計算)。

上記の行の計算では、OIA 行は無視されます。例えば、セッション・ウィンドウが 25 行である場合、24 行のテキストしかありません。

この規則の利点は、セッション・ウィンドウの下部で長方形を指定したい場合、セ ッション・ウィンドウの行が 25 行か、43 行か、50 行かにかかわらず、この計算 は正しい結果が得られることです。同様に、セッション・ウィンドウの右側で長方 形を指定したい場合、セッション・ウィンドウの列が 80 列か、132 列かにかかわ らず、この計算は正しい結果が得られます。

次の表は、いくつかの計算の結果を示しています。

表 7. 行の負の値

行の負の値	24 列のテキストがあ	42 列のテキストがあ	49 列のテキストがあ
	るセッション・ウィ	るセッション・ウィ	るセッション・ウィ
	ンドウ内の実際の値	ンドウ内の実際の値	ンドウ内の実際の値
	(OIA 行は無視)	(OIA 行は無視)	(OIA 行は無視)
-1	24	42	49
-2	23	41	48
-3	22	40	47

表 8. 列の負の値

列の負の値	80 列のセッション・ ウィンドウ内の実際の値	132 列のセッション・ ウィンドウ内の実際の値
-1	80	132
-2	79	131
-3	78	130

この規則を利用するかどうかにかかわらず、座標 (1,1) および (-1,-1) の長方形領域 は、セッション・ウィンドウのテキスト域全体を意味することに注意してくださ い。

第6章 マクロ・ランタイムによるマクロ画面の処理方法

このセクションでは、マクロ・ランタイムがマクロ画面を処理するときに発生する アクティビティーについて説明します。この時点でこのトピックを読むのは煩わし いかもしれませんが、このトピックは重要であり、いずれこのトピックが疑問点に なります。この章の最初の部分(『概要』)を読み終えたら、具体的な疑問が生じる まで、残りの部分をスキップすることもできます。

概要

例として使用するシナリオ

この章では、例として、 13 ページの『第 3 章 単純マクロの記録と再生』で記録 したマクロのシナリオを使用します。このマクロには、2 つのマクロ画面 (Screen1 と Screen2) しか含まれていません。

このシナリオは、マクロ・ランタイムが Screen1 のすべてのアクションを実行し、 次に処理するマクロ画面を検索する準備ができている段階から始まります。

Screen1 は、「ISPF 基本オプション・メニュー」を処理するマクロ画面です (14 ページの図 5 を参照)。 表 9 は、Screen1 の内容の概念視点を示しています。

<screen> エレメント Screen1 に含まれて</screen>	XML エレメントの内容
いる XML エレメント	
<description></description>	ディスクリプター:
	 入力禁止標識がクリアされます (入力は禁止されません)。
<actions></actions>	アクション:
	1. テキスト・カーソルを行 4、列 16 に移 動する。
	2. '3[enter]' と入力する。
<nextscreens></nextscreens>	このマクロ画面の後に表示されるマクロ画面 の名前:
	Screen2

表 9. マクロ画面 Screen1 の内容

Screen2 は、「ユーティリティー選択パネル (Utility Selection Panel)」を処理する マクロ画面です (15 ページの図 6 を参照)。 44 ページの表 10 は、Screen2 の内 容の概念視点を示しています。

表 10. マクロ画面 Screen2 の内容

<screen> エレメント Screen2 に含まれて</screen>	XML エレメントの内容
いる XML エレメント	
<description></description>	ディスクリプター:
	 入力禁止標識がクリアされます (入力は禁止されません)。
	• 80 個のフィールドがあります。
	• 3 つの入力フィールドがあります。
<actions></actions>	アクション (ホスト・アプリケーションは、 正しい入力フィールドにテキスト・カーソル を事前に配置します):
	1. '4[enter]' と入力する。
<nextscreens></nextscreens>	このマクロ画面の後に表示されるマクロ画面 の名前:
	 (なし。これは、このマクロの最後のマクロ面面です。)

マクロ画面の処理ステージ

マクロの再生中に、マクロ・ランタイムは、マクロが終了するまで、同じ3つのステージのアクティビティーを何度もループします。

- 1. 次に処理するマクロ画面を決定する。
- 2. 選択したマクロ画面を新しい現行マクロ画面にする。
- 3. 新しい現行マクロ画面の <actions> エレメントのアクションを実行する。

図 18. マクロ画面の処理ステージ

ステージ 1 の詳細

ステージ 1 には、ステージ 2 や 3 より詳しい説明が必要です。ステージ 1 自体 に次の 3 つのステップがあります。

- 1(a) 候補のマクロ画面の名前を、有効な次画面のリストに追加する。
- 1(b) 画面認識を行って、候補マクロ画面の 1 つを、セッション・ウィンドウに現在表示 されている実際のアプリケーション画面と一致させる。
- 1(c) 候補のマクロ画面の名前を、有効な次画面のリストから除去する。

図 19. ステージ 1 の 3 つのステップ

上記の各ステップには、有効な次画面のリストが必要です。

有効な次画面のリストは、マクロ画面名を保持できるリストです。マクロ・ランタ イムは、マクロ再生の始め (最初のマクロ画面を再生する前) にこのリストを作成 し、マクロ再生が完了した後、このリストを破棄します。最初、このリストは空で す (この章で後述する一時画面を除く)。

マクロの再生中、マクロ・ランタイムが次に処理するマクロ画面を決定する必要が あるたびに、マクロ・ランタイムは、有効な次画面のリストを使用して、3 つのス テップ 1(a)、1(b)、および 1(c) を実行します。

プロセス全体 (3 つの全ステージ)の概要

ステージ 1 では、マクロ・ランタイムは、次に処理するマクロ画面を決定します。 上記のセクションで説明したように、ステージ 1 には 3 つのステップがありま す。

ステップ 1(a) では、マクロ・ランタイムは、現行のマクロ画面の後に表示される可 能性があるマクロ画面の名前を収集し、これらの名前を有効な次画面のリストに追 加します。このリストには、このような画面が 1 つしかない場合もあれば、複数あ る場合もあります。このシナリオの例では、マクロ・ランタイムは、Screen1 の <nextscreens> エレメントを調べ、1 つの名前 (Screen2) を見付け、その名前をリ ストに追加します (43 ページの表 9 を参照)。

ステップ 1(b) では、マクロ・ランタイムは、リスト上の各マクロ画面を定期的に調べて、その画面がアプリケーション画面と一致するかどうかを判別します。

ここでは、時間的要因があります。マクロ・ランタイムが現行のマクロ画面でアク ションを実行したばかりなので (Screen1 で、<actions> エレメントの最後のアクシ ョンとして '3[enter]' を入力)、ホスト・アプリケーションは、セッション・ウィン ドウを変更中です。この結果、古いアプリケーション画面(「ISPF 基本オプショ ン・メニュー」)ではなく、新しいアプリケーション画面(「ユーティリティー選択 パネル (Utility Selection Panel)」)を表示します。しかし、この変更は、即時に行 われるわけではありません。この変更には数百ミリ秒かかり、ホストから数パケッ ト分のデータが必要になる場合があります。

したがって、ステップ 1(b) では、OIA 行またはセッション・ウィンドウの表示ス ペースが更新されるたびに、マクロ・ランタイムは、有効な次画面のリストに指定 されたマクロ画面 (単数または複数) を調べて、それらのいずれかが現行の状態のア プリケーション画面と一致するかどうかを判別します。

最終的に、マクロ・ランタイムがリスト上のマクロ画面のいずれかをアプリケーシ ョン画面と一致させることができるまで、セッション・ウィンドウは更新されま す。

ステップ 1(c) では、マクロ・ランタイムは、有効な次画面のリストからすべてのマ クロ画面名を除去します (一時画面がある場合は、それを除く)。

ステージ 2 では、マクロ・ランタイムは、選択されたマクロ画面 (ステップ 1(b) でアプリケーション画面と一致した画面) を、新しい現行マクロ画面にします。

最後にステージ 3 では、マクロ・ランタイムは、Screen2 の <actions> エレメント のアクションを実行します。

概要の結論

初めて本書をお読みになる場合は、この時点で、この章の残りの部分をスキップし て、次の章に進むことができます。後で、マクロ・ランタイムがマクロ画面を処理 する方法について疑問が生じた場合に、この章に戻って残りの部分をお読みくださ い。

この章の残りの部分では、概要で説明したのと同じ処理手順を説明していますが、 各ステップについてもっと詳しく記載しています。

ステージ 1:次に処理するマクロ画面を決定する

前述のように、ステージ 1 には 3 つのステップがあります。すなわち、有効な次 画面のリストへのマクロ画面名の追加、画面認識の実行、および有効な次画面のリ ストからのマクロ画面名の除去です。

有効な次画面のリストへのマクロ画面名の追加 (ステップ 1(a))

このステップでは、マクロ・ランタイムは、候補のマクロ画面の名前を、有効な次 画面のリストに入れます。

有効な次画面

ホスト・アプリケーションがセッション・ウィンドウにアプリケーション画面を表示しているときに、ユーザー入力が行われた場合、通常、数個のアプリケーション 画面 (多くの場合は、1 つだけ) が次に表示されます。

このシナリオの例では、現行のマクロ画面は Screen1、現行のアプリケーション画 面は「ISPF 基本オプション・メニュー」、入力は「3」と Enter キーです(43 ペ ージの表 9 を参照)。このコンテキストでは、1 つのアプリケーション画面(「ユー ティリティー選択パネル (Utility Selection Panel)」)だけが次に表示されます。し たがって、有効な次画面のリストに追加する必要があるのは、1 つのマクロ画面の 名前 (Screen2) だけです。

しかし、ちょっと待ってください。「ISPF 基本オプション・メニュー」には、約 30 の入力が可能です (15 のオプション、6 つのメニュー選択項目、および 8 つの ファンクション・キー)。リストには、たった 1 つではなく、30 個のマクロ画面名 があるはずです。

有効な次画面のリストに通常、1 つまたは数個の名前しかない理由は、なんらかの 特定のタスクを達成することを目的とする一連の命令をマクロが実行していること です。Screen1 では、その命令の目的は、「ISPF 基本オプション・メニュー」か ら、「ユーティリティー選択パネル (Utility Selection Panel)」に切り替えることで す。この変更を行うために必要なアクションが実行され ('3[enter]')、予想されたア プリケーション画面が表示されるのをマクロ画面は待機します。

マクロ・ランタイムによる候補マクロ画面名の選択方法

このセクションでは、マクロ・ランタイムが、有効な次画面のリストに入れるマク ロ画面名を選択する方法を説明します。次の 2 つの場合があります。

 最初のマクロ画面が再生される場合、マクロ・ランタイムは、入り口画面のマー クが付いている、マクロ内の任意のマクロ画面の名前を選択します。 • 後続のすべてのマクロ画面が再生される場合、マクロ・ランタイムは、現行マクロ画面の <nextscreens> エレメントで検出する名前を使用します。

最初のマクロ画面: マクロの再生が始まるときには、有効な次画面のリストは空で す (一時画面を除く。『一時画面』を参照)。

最初に処理されるマクロ画面の候補を見付けるために、マクロ・ランタイムは、マ クロ全体を検索し、入り口画面のマークが付いている各マクロ画面を見付け、これ らのマクロ画面の名前をリストに追加します。

入り口画面の設定 (<screen> エレメントの属性) は、この目的のために存在し、最 初に処理される画面として表示できるマクロ画面にマークを付けます。

マクロが記録されるときに、マクロ・オブジェクトは、デフォルトで、最初に記録 されるマクロ画面だけに入り口画面のマークを付けます。記録が完了した後、マク ロ開発者は、任意のマクロ画面に入り口画面のマークを付けたり、マークを解除し たりすることができます。複数の入り口画面が存在できます。

入り口画面については、 139 ページの『入り口画面』で詳しく説明しています。

入り口画面のマークが付いているマクロ画面がない場合、マクロ・ランタイムは、 マクロ内のすべてのマクロ画面を、最初に処理されるマクロ画面の候補として使用 します。

後続のマクロ画面: 後続のマクロ画面 (最初のマクロ画面の直後の画面を含む) の 場合、マクロ・ランタイムは、現行マクロ画面の <nextscreens> エレメントにリス トされる候補マクロ画面の名前を見付けます。

このシナリオの例では、Screen1 が現行のマクロ画面であり、その <nextscreens> エレメントには、1 つのマクロ画面の名前 (Screen2) が含まれています (43 ペー ジの表 9 を参照)。したがって、マクロ・ランタイムは、リストに Screen2 を追加 します。

どんなに多くのマクロ画面名が <nextscreens> エレメントにリストされていても、 マクロ・ランタイムは、すべての画面名を有効な次画面のリストに追加します。

マクロの記録時に、マクロ・オブジェクトが新しいマクロ画面の記録を開始する と、その新しいマクロ画面の名前 (例えば、Screen2) を、記録が終了したばかりの マクロ画面 (Screen1) の <nextscreens> エレメントに保管します。したがって、記 録されたマクロの各マクロ画面 (最後の画面を除く) は、<nextscreens> エレメント に保管されている 1 つのマクロ画面の名前を持ちます。

それ以降、マクロ開発者は、マクロ内の任意のマクロ画面の名前を、任意のマクロ 画面の <nextscreens> エレメントに追加したり、そのエレメントから削除したりす ることができます。

<nextscreens> エレメントの詳細については、 137 ページの『有効な次画面』で説 明しています。

ー時画面: 一時画面は、マクロの任意のポイントで生じる画面であり、予期せず発 生し、常にクリアする必要がある画面です。一時画面の例は、無効な入力に応答し て表示されるエラー画面です。 マクロ・オブジェクトは、マクロの記録時にマクロ画面に一時画面のマークを付け ることはありません。しかし、それ以降、マクロ開発者は、任意のマクロ画面に一 時画面のマークを付けることができます。

マクロの再生が開始すると、マクロ・ランタイムはマクロを検索し、一時画面のマ ークが付いている各マクロ画面を見付け、各一時マクロ画面の名前を有効な次画面 のリストに追加します。これらの名前は、マクロ再生中、リストに存在します。

一時画面の詳細については、140ページの『一時画面』を参照してください。

画面認識 (ステップ 1(b))

このステップでは、マクロ・ランタイムが、有効な次画面のリストに名前が指定されているマクロ画面の 1 つを、現行のアプリケーション画面と一致させます。

このプロセスは、画面認識と呼ばれます。これは、マクロ・ランタイムが、リスト 上のマクロ画面の 1 つを、セッション・ウィンドウに現在表示されているアプリケ ーション画面に対応するものとして認識するからです。

評価の概要

マクロ・ランタイムは、候補マクロ画面を、有効な次画面のリスト内にそれらの名 前が表示されている順に評価します。

マクロ・ランタイムが、候補の 1 つがアプリケーション画面と一致することを検出 すると、ただちに評価を停止し、リストから候補名を除去する次のステップ (ステ ップ 1(c)) に進みます。一致する画面は、次に処理されるマクロ画面になります (ス テージ 2)。

しかし、マクロ・ランタイムがリストに名前が指定された各マクロ画面を評価し て、一致を検出しない場合、マクロ・ランタイムは、評価を一時的に停止し、セッ ション・ウィンドウが更新されるまでは何も実行しません。

評価のやり直し

マクロ・ランタイムが画面認識を続ける間、ホスト・アプリケーションは、新しい アプリケーション画面でセッション・ウィンドウの更新を続けます。このシナリオ の例では、ホスト・アプリケーションは、セッション・ウィンドウを更新して、 「ユーティリティー選択パネル」を表示します(43 ページの表 9 および 44 ペ ージの表 10 を参照)。このプロセスには数百ミリ秒かかり、ホストから数パケット 分のデータが必要になる場合があります。

この状況は、画面が更新されるまでマクロ・ランタイムが画面認識を一時的に停止 する理由の説明になります。画面認識が失敗した場合、新しいアプリケーション画 面が不完全であることが理由である可能性があります。したがって、マクロ・ラン タイムは待機します。

OIA 行が更新されるか、セッション・ウィンドウの表示スペースが更新されるたび に、マクロ・ランタイムは有効な次画面のリストを調べ、現行のアプリケーション 画面との一致を見付けようとします。一致が見付からない場合、マクロ・ランタイ ムは再び待機します。 マクロ・ランタイムは、画面認識が成功するまで、待機と再評価のサイクルを複数 回実行する場合があります。

最終的に、必要な新しいアプリケーション画面に到達し、マクロ・ランタイムは、 リストに名前が指定されたマクロ画面の 1 つを、新しいアプリケーション画面と一 致させることができます。

マクロ画面がアプリケーション画面と一致するかどうかの判別

マクロ・ランタイムは、マクロ画面内の個々のディスクリプターを現行のセッショ ン・ウィンドウと比較することによって、マクロ画面が現行のアプリケーション画 面と一致するかどうかを判別します。

このシナリオの例では、マクロ・ランタイムは、有効な次画面のリストで名前 Screen2 を見付け、Screen2 を取り出し、そのディスクリプターを調べ、セッショ ン・ウィンドウと比較します。

各マクロ画面には、1 つの <description> エレメントが含まれ、そのエレメント自体に 1 つ以上のディスクリプターが含まれています。ディスクリプターは、true か false にすることができる、セッション・ウィンドウ (現在の状態のアプリケーション画面) についての事実のステートメントです。このシナリオの例では、Screen2 には次の 3 つのディスクリプターが含まれています。

- 入力禁止標識がクリアされます (入力は禁止されません)。
- セッション・ウィンドウには 80 個のフィールドがあります。
- セッション・ウィンドウには3つの入力フィールドがあります。

<description> エレメントに複数のディスクリプターがある場合、マクロ・ランタイ ムがディスクリプターを評価し (ブール true または false として)、それらの結果 を 1 つの結果 (true または false) に結合するために使用するメソッドは、本書で は記述されていない追加の構成情報によって決まります。

しかし、このシナリオの例では、Screen2 はデフォルトで構成されるので、マク ロ・ランタイムは 3 つのディスクリプターをそれぞれ順に評価します。 3 つのす べてが true である場合、マクロ・ランタイムは、全体的な結果が true であり、 Screen2 が現行のアプリケーション画面と一致するという結論を出します。

ディスクリプターの評価の詳細については、 58 ページの『ディスクリプターの評価』を参照してください。

2 つの認識機能

画面認識のタイムアウト設定: タイマーが切れる前に画面認識が行われないとマク ロ・ランタイムがマクロを終了させる、タイムアウト値を設定できます (142 ペー ジの『画面認識のタイムアウト設定』を参照)。

認識限界: マクロ・ランタイムが認識カウントに相当する回数、マクロ画面 (例え ば、ScreenA) を認識すると、マクロ・ランタイムにマクロを終了させるか、指定さ れたマクロ画面にジャンプさせる認識カウントを設定できます (143 ページの『認 識限度 (「画面 (Screens)」タブの「一般 (General)」タブ)』を参照)。

有効な次画面のリストからの候補マクロ画面の名前の除去 (ステップ 1(c))

画面認識が成功した後、マクロ・ランタイムは、ただちに次のタスクを開始しま す。このタスクは、有効な次画面のリストのクリーンアップです (ステップ 1(c))。

これは単純なステップです。マクロ・ランタイムは、認識しているかどうかにかかわらず、すべての候補マクロ画面の名前を有効な次画面のリストから除去します。

リストに一時画面の名前が含まれている場合、それらの名前はリストに残ります (140 ページの『一時画面』を参照)。

ステージ 2: 選択された候補を新しい現行マクロ画面にする

ステージ 2 は単純です。ステージ 2 では、マクロ・ランタイムは、選択された候 補を新しい現行マクロ画面にします。

このシナリオの例では、マクロ・ランタイムが、Screen2 を新しい現行マクロ画面 にします。セッション・ウィンドウは、新しいアプリケーション画面(「ユーティ リティー選択パネル」)を表示します(43 ページの表 9 および 44 ページの表 10 を参照)。

マクロ・ランタイムは、ただちにステージ3を開始します。

ステージ 3: 新しい現行マクロ画面のアクションを実行する

ステージ 3 では、マクロ・ランタイムは、新しい現行マクロ画面の <actions> エ レメント内のアクションを実行します。新しい現行マクロ画面に <actions> エレメ ントが含まれていない場合、または <actions> エレメントが空である場合、マク ロ・ランタイムはこのステージをスキップします。

各マクロ画面には、通常、実行される 1 つ以上のアクションを含む <actions> エ レメントが含まれています。アクションは、なんらかのアクティビティー (セッシ ョン・ウィンドウへの一連のキーの送信、ポップアップ・ウィンドウでのユーザー へのプロンプトの表示、画面からのテキスト・ブロックの取り込みなど)を起こす 命令です。

このシナリオの例では、Screen2 には次の 1 つのアクションしか含まれていません。

• 「4」と入力し、Enter キーを押す。

Screen2 には、正しい入力フィールドにテキスト・カーソルを置くアクションは必要ありません。これは、「ユーティリティー選択パネル」が自動的にテキスト・カーソルをその位置に置くからです。

<actions> エレメントに複数のアクションが含まれている場合、マクロ・ランタイムは、<actions> エレメントに指定される順に、各マクロ・アクションを実行します。

アクションの詳細については、 77 ページの『第 8 章 マクロ・アクション』を参照してください。

アクション後の遅延の挿入

マクロ・ランタイムは人間のユーザーよりはるかに迅速にアクションを実行するの で、マクロの再生中に予測しない問題が発生し、アクションが予想通りに実行され ない可能性があります。この原因は、前のアクションへの依存関係が生じることで す。

このタイプの問題を避けるために、マクロ・ランタイムは、各マクロ画面のすべて の入力アクションやプロンプト・アクションの後に 150 ミリ秒の遅延を、すべての マクロ画面の最後のアクションの後に 300 ミリ秒の遅延をデフォルトで挿入します (145 ページの『アクション間の休止 (「マクロ (Macro)」タブ)』を参照)。

この機能を使用可能なままにしてください。ただし、必要な場合は使用不可にする ことができます。遅延を 150 ミリ秒および 300 ミリ秒から他の値に変更できま す。

必要に応じて、特定のマクロ画面の遅延期間を変更することもできます (146 ページの『休止時間の設定 (「画面 (Screens)」タブの「一般」タブ)』を参照)。

また、特定のアクションに対して、アクション後に休止アクションを追加すること によって、遅延を増やすこともできます (107 ページの『休止アクション (<pause> エレメント)』を参照)。

処理サイクルの繰り返し

マクロ・ランタイムが現行マクロ画面の <actions> エレメント内のすべてのアクションを実行した後、マクロ・ランタイムは、新しい現行マクロ画面の <nextscreens> エレメントにリストされている候補マクロ画面を使用して、ステップ 1(a) からただちに処理サイクルを再開します。

マクロの終了

マクロ・ランタイムは、出口画面のマークが付いているマクロ画面の処理を終了す ると、マクロを終了します。

このシナリオの例では、Screen2 に出口画面のマークが付けられています (44 ページの表 10 を参照)。

出口画面の設定 (<screen> エレメントの属性) は、この目的のために存在し、マク ロを終了するマクロ画面にマークを付けます。

マクロが記録されるときに、マクロ・オブジェクトは、デフォルトで、最後に記録 されるマクロ画面に出口画面のマークを付けます。記録が完了した後、マクロ開発 者は、任意のマクロ画面に出口画面のマークを付けたり、マークを解除したりする ことができます。複数の出口画面が存在できます。

出口画面については、140ページの『出口画面』で詳しく説明しています。

第7章 画面記述と画面認識

この章では、次の内容を取り上げます。

- ディスクリプター、画面認識、および画面記述という用語
- 「記述 (Description)」タブ
- Macro Facility がアプリケーション画面の記述を記録する方法
- 複数のディスクリプターを結合する方法
- 各種タイプのディスクリプター

用語の定義

ディスクリプターとは、マクロ画面の <description> エレメント内にある XML エ レメントであり、マクロ画面と対応するアプリケーション画面の識別特性を指定し ます。

例えば、ScreenB という名前のマクロ画面に、アプリケーション画面の行 3 にスト リング ISPF Primary Option Menu が入っていることを示すストリング・ディスク リプター (<string> エレメント) が含まれているとします。マクロの再生時に、マ クロ・ランタイムが次に処理するマクロ画面を決定するとき、および ScreenB が候 補である場合、マクロ・ランタイムは、ScreenB のディスクリプターを実際のアプ リケーション画面と比較します。そのディスクリプターが実際のアプリケーション 画面と一致する場合 (アプリケーション画面の行 3 に、そのストリングが含まれて いる場合)、マクロ・ランタイムは、次に処理するマクロ画面として ScreenB を選 択します。

画面認識とは、マクロ・ランタイムが候補マクロ画面を現行アプリケーション画面 と一致させようとするときに実行するプロセスです。

43 ページの『第6章マクロ・ランタイムによるマクロ画面の処理方法』で説明 したように、マクロ・ランタイムは、次に処理されるマクロ画面を判別する必要が ある場合、候補マクロ画面の名前(通常、現行マクロ画面の <nextscreens> エレメ ントで検出される)を、有効な次画面のリストに入れます。次に、ホスト・アプリ ケーションがセッション・ウィンドウを新しいアプリケーション画面で更新するの で、マクロ・ランタイムは、リスト上の各マクロ画面のディスクリプターを新しい アプリケーション画面と比較します。最終的に、マクロ・ランタイムがリスト上の マクロ画面のいずれかをアプリケーション画面と一致させることができる(例え ば、ストリング ISPF Primary Option Menu が行3に表示される)まで、アプリケ ーション画面が更新されます。一致したマクロ画面は、次に処理されるマクロ画面 になります(45 ページの『プロセス全体(3 つの全ステージ)の概要』を参照)。

画面記述とは、マクロ画面の <description> エレメントにディスクリプターを追加 するプロセスです。画面記述を行うのは、マクロ画面の記述タブに進み、ディスク リプター (上記の例のストリング・ディスクリプターなど) を作成または編集する場 合です。同様に、マクロ・オブジェクトは、マクロの記録時に、作成する新しいマ クロ画面ごとに 1 つ以上のディスクリプターを作成します (56 ページの『記述の 記録』を参照)。

記述タブの概要

記述タブの例

マクロ・エディターの「画面 (Screens)」タブ上の「記述 (Description)」タブで は、マクロ画面の <description> エレメント内に保管されている情報にアクセスで きます。図 20 は、記述タブの例を示しています。

ホスト・アクセス・マクロ・エディター	X
マクロ 画面 リンク 変数	
雨西夕 Caraon2 画面去剃吟	
ディスクリプター Field Counts and OLA	
Field Counts and OIA	
<new descriptor="" string=""></new>	
<new attribute="" descriptor=""></new>	
<new condition="" descriptor=""> </new>	
フィー= ピ粉	
オプショナル false ・ 逆ディスクリプター false ・	
OIA が禁止解除になるのを待つ True オフショナル false マ	
ホスト ID	
	4
一般的な画面特性でこの画面を認識する	

図 20. 記述タブ

前の図では、マクロ・エディターの「画面 (Screens)」タブが選択されています。現 在選択されている画面の名前 Screen2 が、画面タブの一番上の「画面名 (Screen Name)」フィールドに表示されています。「画面名 (Screen Name)」フィールドの 下には、「一般 (General)」、「記述 (Description)」、および「アクション (Actions)」のサブタブがあります。記述タブが選択されています。

上記の図の記述タブを見ると、上部領域と下部領域があることが分かります。

上部領域には、単一のディスクリプター・エレメントを全体と見なして作用するコ ントロールがあります。特に、記述タブの左上隅にある「ディスクリプター (Descriptor)」リスト・ボックスには、現在選択されているディスクリプターの名前 が入っています。上記の図では、現在選択されているディスクリプターは、リスト の一番上にある Field Counts and OIA ディスクリプターです。 (ディスクリプターには名前がありません。 Field Counts and OIA は、ディスクリプターのタイプです。)

記述タブの下部領域には、現在選択されているディスクリプターの内容が表示され ます。現在選択されているディスクリプターは「Fields Counts and OIA」ディス クリプターなので、記述タブの下部は、そのタイプのディスクリプターに適した内 容を提示します。ユーザーが別のタイプのディスクリプター (例えば、ストリン グ・ディスクリプター)を作成し、選択した場合、下部領域には、ストリング・デ ィスクリプターに該当する内容が提示されます。

54 ページの図 20 の記述タブの下部領域をもっと詳しく調べると、「Field Counts and OIA」ディスクリプターには、次の 3 つの ID テストが含まれている ことが分かります。

- ・ 画面には 80 個のフィールドがある (「フィールド数 (Number of Fields)」フィ ールドが 80 に設定されている)
- ・ 画面には 3 つの入力フィールドがある (「入力フィールド数 (Number of Input Fields)」フィールドが 3 に設定されている)
- ・ 画面の入力禁止標識がクリアされている(「OIA が禁止解除になるのを待つ (Wait for OIA to Become Uninhibited)」リスト・ボックスが true に設定され ている)。

マクロ・ランタイムは、このマクロ画面をアプリケーション画面に一致させようと するときに、これらの 3 つの ID テストを適用します。

注意: マクロ・エディターは「Fields Counts and OIA」ディスクリプターを、3 つ のテストが含まれている 1 つのディスクリプターとして提示しますが、実際には、 マクロ言語が、これらの 3 つのテストを別々の独立した 3 つのディスクリプター として定義します。 63 ページの『「Field Counts and OIA」ディスクリプター』 を参照してください。

54 ページの図 20 の記述タブの下部領域は、「Field Counts and OIA」ディスク リプターの 3 つのテストのそれぞれに、「オプショナル (Option)」と「逆ディス クリプター (Inverse Descriptor)」というラベルが付いた 2 つのフィールドも表 示します。今のところ、これらの 2 つのフィールドを無視できます。これらのフィ ールドについては、 59 ページの『デフォルト結合メソッド』で説明します。

新しいディスクリプターの作成

54 ページの図 20 の「ディスクリプター」リスト・ボックスを再度調べると、最 初の項目だけが実際のディスクリプターであることに気付くはずです。残りの選択 項目は、すべて不等号括弧で囲まれ、new という語で始まっています。これらの選 択項目は、新しいディスクリプターを作成するためのものです。 54 ページの図 20 内のリストは次のとおりです。 Fields Counts and OIA <new string descriptor> <new cursor descriptor> <new attribute descriptor> <new condition descriptor> <new variable update>

図 21. 実際のディスクリプターが 1 つある「ディスクリプター」リスト・ボックスの内容

例えば、<new string descriptor> をクリックした場合、マクロ・オブジェクトは、 新しいストリング・ディスクリプターを作成し、リストの先頭に置きます。記述タ ブの下部領域では、ストリング・ディスクリプターに属するさまざまなフィールド (例えば、行と列の位置や、文字ストリング) に入力できます。「ディスクリプタ ー」リスト・ボックスは次のようになります。

String descriptor(3, 29) Fields Counts and OIA <new string descriptor> <new cursor descriptor> <new attribute descriptor> <new condition descriptor> <new variable update>

図 22. 実際のディスクリプターが 2 つある「ディスクリプター」リスト・ボックスの内容

上記の図では、現在選択されているディスクリプターは、リストの一番上にある String descriptor です (3,29 は、行 3、列 29 を表します)。 Field Counts and OIA ディスクリプターは、リスト上の 2 番目になりました。

上記の図のように、マクロ・ランタイムが複数のディスクリプターを処理する方法 については、 58 ページの『ディスクリプターの評価』を参照してください。

記述の記録

記録される情報

マクロの記録時に、マクロ・オブジェクトは、作成する新しい各マクロ画面の新しい くdescription> エレメントに、1 つ以上のディスクリプターを追加します。

マクロのマクロ画面が最初に記録される場合、マクロ・オブジェクトは、次の内容 を指定した 1 つのディスクリプター (「Field Counts and OIA」ディスクリプタ ー) だけを作成します。

- 「フィールド数 (Number of Fields)」は、ブランクに設定される (つまり、フィールド数を無視する)。
- 「入力フィールド数 (Number of Input Fields)」は、ブランクに設定される (つまり、入力フィールド数を無視する)。
- 「OIA が禁止解除になるのを待つ (Wait for OIA to Become Uninhibited)」フ ィールドは、true に設定される。

したがって、記録されたマクロが(改訂なく)再生されるときに、マクロ・ランタイムは、入力禁止標識がクリアされているかどうかに基づいて、最初のマクロ画面を対応するアプリケーション画面に一致させます。

最初のアプリケーション画面以降のマクロのすべてのアプリケーション画面に対し て、マクロ・オブジェクトは、同様に 1 つのディスクリプター (「Field Counts and OIA」ディスクリプター) だけを作成しますが、次のように内容が異なりま す。

- 「フィールド数 (Number of Fields)」は、アプリケーション画面内の実際のフィールド数に設定される (0 の場合がある)。
- 「入力フィールド数 (Number of Input Fields)」は、アプリケーション画面内の 実際の入力フィールド数に設定される (0 の場合がある)。
- 「OIA が禁止解除になるのを待つ (Wait for OIA to Become Uninhibited)」フ ィールドは、true に設定される。

したがって、記録されたマクロが(改訂なく)再生されるときに、マクロ・ランタイムは、入力禁止標識がクリアされているかどうか、マクロ画面の記述内のフィール ド数がアプリケーション画面内のフィールド数と一致するかどうか、およびマクロ 画面の記述内の入力フィールド数が、アプリケーション画面内の入力フィールド数 と一致するかどうかに基づいて、最初のマクロ画面以降のすべてのマクロ画面を、 対応するアプリケーション画面に一致させます。

記録された記述が機能する理由

記録された記述は、少なくとも次の3つの理由で適切に働きます。

第一は、「Field Counts and OIA」ディスクリプターの 3 つの部分が、どのアプ リケーション画面にも確実に適用できることです。つまり、どのアプリケーション 画面にも、いくつかのフィールド (おそらく、その数は 0)、いくつかの入力フィー ルド (おそらく、0)、および入力禁止標識 (設定またはクリア) があります。

第二に、「フィールド数 (Number of Fields)」ディスクリプターと「入力フィール ド数 (Number of Input Fields)」ディスクリプターの組み合わせにより、アプリケ ーション画面の記述の信頼性が高まることです。これは、アプリケーション画面に は、一般に多数のフィールドが含まれているからです。例えば、 15 ページの図 6 に表示されている「ユーティリティー選択パネル」には、現在、すべてのタイプの 80 個のフィールドが含まれています。その内の 3 つは入力フィールドです。 14 ページの図 5 に表示されている「ISPF 基本オプション・メニュー」には、現在、 すべてのタイプの 116 個のフィールドが含まれています。その内の 3 つは入力フ ィールドです。アプリケーション画面に多数のフィールドが含まれている場合、各 画面に同数のフィールドと同数の入力フィールドがあるので、マクロ・ランタイム が 2 つのアプリケーション画面を混合する可能性が低くなります。

第三に、おそらくこれが最も重要ですが、画面認識時に、マクロ・ランタイムは、 新しいアプリケーション画面を、有効な次画面と呼ばれる、マクロ画面の短いリス ト (通常、非常に短いリスト)と比較することです(44ページの『ステージ1の 詳細』を参照)。したがって、1つのマクロ画面を、マクロ内の他のすべてのマクロ 画面と区別する必要はありません。有効な次画面のリスト内の他の画面と区別する だけです。多くの場合、このリストは、1つのマクロ画面から構成されます。

記録されたディスクリプターが提供するフレームワーク

マクロの記録は、マクロのフレームワークをすばやく提供するので、非常に便利な 機能です。しかし、一部のマクロ画面の場合、記録された記述では、マクロ・ラン タイムが 1 つのアプリケーション画面を別の類似したアプリケーション画面と確実 に区別することができない場合があります。このような場合は、記録された記述を 改善する必要があります。

多くの場合、記録された記述を改善する最も簡単な方法は、ストリング・ディスク リプターを追加することです。例えば、マクロ画面が、 15 ページの図 6 に表示さ れている「ユーティリティー選択パネル」用である場合、アプリケーション画面の 行 3 にストリング 'Utility Selection Panel' が含まれていることを指定するスト リング・ディスクリプターを追加できます。もちろん、ストリング・ディスクリプ ターの使用に限定されません。一部の状況では、アプリケーション画面が正しく認 識されることを確実にするために、1 つ以上の他のディスクリプター (例えば、カ ーソル・ディスクリプター、属性ディスクリプター、または条件ディスクリプター) を使用する必要があります。

ディスクリプターの評価

ここでは、マクロ画面がアプリケーション画面と一致するかどうかをマクロ・ラン タイムが判別する方法を詳しく説明します。

役立つ情報

以下のサブセクションを読む前に、次の点に注意してください。

- 大部分のマクロ画面では、<description> エレメントに複数のディスクリプター が含まれています。(「Field Counts and OIA」ディスクリプターには、最高 3 つの独立したディスクリプターを含むことができることに注意してください。
 63 ページの『「Field Counts and OIA」ディスクリプター』を参照してください。)
- 記述タブ内のデフォルト設定では、すべてのディスクリプターが必須であり(各 ディスクリプターの「オプショナル (Optional)」設定は false)、デフォルトの結 合規則が使用されます。
- 最も一般的なシナリオでは、すべてのディスクリプターが必須です。(つまり、3 つのディスクリプターを定義した場合、マクロ画面が認識されるために、それらの3つ全部を true にします)。このシナリオを使用する場合は、デフォルト設定を使用してください。
- デフォルトのシナリオよりも複雑なシナリオに取り組む場合は、uselogic メソッドを使用してください。

プロセスの概要

プロセスの概要は次のとおりです。

- 1. マクロ・ランタイムは、各ディスクリプターを個々に評価し、そのディスクリプ ターのブール結果 (true または false) を得ます。
- 2. マクロ・ランタイムは、個々のディスクリプターのブール結果を結合して、記述 全体が true であるか (マクロ画面がアプリケーション画面と一致する)、false
であるかを判別します。個々のディスクリプターの結果を結合するために、マク ロ・ランタイムは、デフォルトの結合メソッドまたは uselogic メソッドを使用 します。

- a. デフォルトの結合メソッドを使用する場合
 - 1) マクロ・ランタイムは、「逆ディスクリプター (Inverse Descriptor)」 オプションが true に設定されているディスクリプターのブール結果を 逆転させる。
 - マクロ・ランタイムは、次のものを使用して個々のディスクリプターの ブール結果を結合する。
 - ディスクリプターごとの「オプショナル (Optional)」オプションの設定
 - ディスクリプターを結合するためのデフォルト規則
- b. uselogic メソッドを使用する場合
 - マクロ・ランタイムは、「逆ディスクリプター」と「オプショナル」の 設定を無視する。
 - マクロ・ランタイムは、uselogic 属性でユーザーが指定する規則を使用して、個々のディスクリプターの結果を結合する。

個々のディスクリプターの評価

マクロ記述内の個々のディスクリプターごとに、マクロ・ランタイムはディスクリ プターを評価し、ブール結果 (true または false) を得ます。

例えば、ディスクリプターがストリング・ディスクリプターである場合、マクロ・ ランタイムは、そのディスクリプターが指定する行と列でアプリケーション画面を 調べ、その位置にあるストリングを、ディスクリプターが指定するストリングと比 較します。この 2 つのストリングが一致する場合、マクロ・ランタイムは、値 true をストリング・ディスクリプターに割り当てます。この 2 つのストリングが 一致しない場合、マクロ・ランタイムは、値 false をストリング・ディスクリプタ ーに割り当てます。

通常、マクロ画面には複数のディスクリプターが含まれています。

しかし、マクロ画面に 1 つのディスクリプターしかない (そのディスクリプターの 「逆ディスクリプター」オプションが true に設定されていないことを前提とする) 場合、その 1 つのディスクリプターが true であれば、記述全体も true であり、 マクロ・ランタイムは、マクロ画面をアプリケーション画面に一致するものと認識 します。一方、その 1 つのディスクリプターが false である場合、記述全体が false であり、マクロ画面は認識されません。

デフォルト結合メソッド

<description> エレメントに複数のディスクリプターがある場合、このセクションで 説明されているデフォルト結合メソッドを使用するか、 61 ページの『uselogic 属 性』で説明されている uselogic 属性を使用する必要があります。

デフォルト結合メソッドの用途

デフォルト結合メソッドが適しているのは、次の 2 つのシナリオだけです。

- 個々のディスクリプターがすべて true である場合だけ、記述全体を true にする (最も一般的なシナリオ)
- 個々のディスクリプターの少なくとも 1 つが true である場合、記述全体を true にする

デフォルト結合メソッドの働きを十分理解している場合を除いて、他のシナリオに デフォルトのメソッドを使用しないでください。

デフォルト結合メソッドは、次のものを使用します。

- 個々のディスクリプターのブール結果(59ページの『個々のディスクリプターの評価』を参照)
- 各個別ディスクリプターの「逆ディスクリプター」オプションの値
- 各個別ディスクリプターの「オプショナル」オプションの値
- デフォルト結合規則

逆ディスクリプター

どのディスクリプターにも、false (デフォルト) か true に設定される「逆ディスク リプター (Inverse Descriptor)」オプションがあります。「逆ディスクリプター」オ プションは、 54 ページの図 20 の「フィールド数 (Number of Fields)」入力フィ ールドの右下にリスト・ボックスとして表示されています。マクロ言語は、ディス クリプター・エレメントの invertmatch 属性を使用して、このオプションを保管し ます。

デフォルトでは、このオプションは false です。したがって、ディスクリプターの 評価には影響ありません。

この設定値が true である場合、マクロ・ランタイムは、ディスクリプターの評価 から得るブール結果を逆転させます。つまり、true を false にするか、false を true にします。

例えば、ストリング・ディスクリプターが true である (ディスクリプター内のスト リングが、アプリケーション・ウィンドウの画面と一致する) にもかかわらず、ス トリング・ディスクリプターの「逆ディスクリプター」オプションが true に設定 されているとマクロ・ランタイムが判断する場合、マクロ・ランタイムは、ストリ ング・ディスクリプターの結果を true から false に変えます。

オプショナル

どのディスクリプターにも、false (デフォルト) か true に設定される「オプショナ ル (Optional)」オプションがあります。このオプションは、 54 ページの図 20 の 「フィールド数 (Number of Fields)」入力フィールドの下にリスト・ボックスとし て表示されています。マクロ言語は、ディスクリプター・エレメントの optional 属性を使用して、このオプションを保管します。

「オプショナル」オプションは、マクロ・ランタイムがデフォルト結合規則を使用 してディスクリプターのブール結果を結合する場合に、個々のディスクリプターの 結果を取り扱う方法を指定します。デフォルトでは、このオプションは、false に設 定されます。これは、ディスクリプターの結果が、オプショナルではなく必須であ ることを示します。

デフォルト結合規則

前述のように、デフォルト結合規則が適しているのは、次の 2 つのシナリオだけで す。

- 個々のディスクリプターがすべて true である場合だけ、記述全体を true にする (最も一般的なシナリオ)
- 個々のディスクリプターの少なくとも 1 つが true である場合、記述全体を true にする

すべてのディスクリプターが true である場合だけ、記述全体を true にしたい場合 は、記述内のすべてのディスクリプターの「オプショナル」設定値を false (デフォ ルト設定) に設定してください。

一方、ディスクリプターの少なくとも 1 つが true である場合に、記述全体を true にしたい場合は、記述内のすべてのディスクリプターの「オプショナル」設定値を true に設定してください。

デフォルト結合規則とその意味を十分に理解している場合を除いて、1 つのマクロ 画面に複数のディスクリプターがあるその他のシナリオで、デフォルト結合規則を 使用しないでください。詳しくは、 267 ページの『1 つのマクロ画面内の複数ディ スクリプターのデフォルト結合規則』を参照してください。

また、規則とその意味を十分に理解している場合を除いて、1 つのマクロ画面内の 複数のディスクリプターの「オプショナル」に異なる設定値 (true と false の混合) を設定しないでください。

uselogic 属性

<description> エレメントの uselogic 属性を使用すると、上記のセクションで説明 したデフォルト結合メソッドで使用可能な複数のディスクリプター間で、もっと複 雑な論理関係を定義できます。

uselogic 属性を使用する場合、マクロ・ランタイムは、個々のディスクリプターの 「逆ディスクリプター」設定と「オプショナル」設定を無視します。

コード・エディターを使用して、手作業で uselogic 属性を <description> エレメ ントに追加する必要があります。マクロ・エディターは、これを制御できません。

uselogic 属性の値は、単純化された論理式であり、その条件は、後続のディスクリ プターの 1 ベースのインデックスです。図 23 は、uselogic 属性を含む <description> エレメントの例を示しています (分かりやすくするために、<string> エレメントの属性の一部が省略されています)。

図 23. <description> エレメントの uselogic 属性の例

上記の図では、uselogic 属性の値は、次のとおりです。

(1 and 2) or (!1 and 3)

この論理式は、正規の論理式ではなく (39 ページの『条件演算子と論理演算子および式』を参照)、uselogic 属性のみで使用される、単純化されたスタイルの論理式です。このスタイルの論理式の規則は、次のとおりです。

- 数表示 1、2、3 などは、それぞれ <description> エレメント内の第 1、第 2、 および第 3 のディスクリプターのブール結果を表します (上記の図では、
 <ia>、 <string>、および <cursor>)。対応するディスクリプターが存在する任意の数表示を使用できます。例えば、<description> エレメントに 7 つのディス クリプターがある場合、7 番目のディスクリプターのブール結果を参照するには 7 を使用し、6 番目のディスクリプターのブール結果を参照するには 6 を使用 し、以下同様です。
- 次の論理演算子だけが使用できます。

表 11. uselogic 属性の論理演算子

演算子:	意味:
and	論理 AND
or	論理 OR (包含)
!	論理 NOT (否定)

- 条件をグループ化するには、小括弧 () を使用できます。
- 以下のエンティティーは使用できません。
 - 算術演算子および式
 - 条件演算子および式
 - 変数
 - Java メソッドの呼び出し

61 ページの図 23 の例では、マクロ・ランタイムは、次の場合、記述全体が true であると判断します。

- 最初のディスクリプターの結果が true であり、2 番目のディスクリプターの結 果が true である (1 および 2)。
- 最初のディスクリプターの結果が false であり、3 番目のディスクリプターの結果が true である (!1 および 3)。

uselogic 属性を使用する場合、マクロ・ランタイムは、個々のディスクリプターの 「逆ディスクリプター」設定と「オプショナル」設定を無視することに注意してく ださい。

ディスクリプター

概要

各タイプのディスクリプターは、1 つのマクロ画面の <description> エレメント内 にある個々の XML エレメントとして保管されます。 初めは、すべてのタイプのディスクリプターを理解する必要はありません。次の 2 つのタイプを理解することから始めてください。

- 「Fields Counts and OIA」ディスクリプター (実際には、3 つの個別ディスク リプターが含まれます)
- ストリング・ディスクリプター

これらのタイプのディスクリプターだけで、多くの、おそらく大部分のアプリケー ション画面を確実に記述できます。しかし、これらのタイプでは十分ではない場合 は、他のタイプのディスクリプターのいずれかを利用する必要があります。

表 12 は、すべてのタイプのディスクリプターをリストし、1 つのマクロ画面 (具 体的には、1 つの <screen> エレメントに属する 1 つの <description> エレメン ト) に存在できる、各タイプのディスクリプター数を示しています。

表 12. ディスクリプターのタイプ、各タイプで使用できる数

ディスクリプターのタイプ:	マクロ画面ごとに使用できるこのタイプの	
	ディスクリプター数:	
Field Counts and OIA	1 (必須)	
ストリング・ディスクリプター	0 またはそれ以上	
カーソル・ディスクリプター	0 または 1	
属性ディスクリプター	0 またはそれ以上	
条件ディスクリプター	0 またはそれ以上	

以下のサブセクションでは、各タイプのディスクリプターを詳しく説明していま す。

「Field Counts and OIA」ディスクリプター

必須

「Field Counts and OIA」ディスクリプターは必須であり、固有のディスクリプターでなければなりません。つまり、どの記述タブ (<description> エレメント) に も、1 つの「Field Counts and OIA」ディスクリプターが含まれていなければなり ません。

次の理由で、この事実が実際に問題を起こすことはありません。

- 「Field Counts and OIA」ディスクリプター自体は必須ですが、それに含まれている3つのテストの内、1つだけが必須です。したがって、実際の要件は、どの記述タブにも1つの「OIA が禁止解除になるのを待つ (Wait for OIA to Become Uninhibited)」ディスクリプターが含まれていなければならないことです。
- マクロ・エディターとコード・エディターがこれらの規則を実行するので、誤っ て複数の「Field Counts and OIA」ディスクリプターが記述タブ(または <description> エレメント)に含まれることがなくなります。例えば、次のとお りです。
 - 「Field Counts and OIA」ディスクリプターを削除しようとすると、記述タ ブの「削除 (Delete)」ボタンは無効です。
 - 記述タブの「ディスクリプター」リスト・ボックスには、「Field Counts and OIA」ディスクリプターの <new> 項目が含まれません。

3 つの別々の独立したディスクリプターを 1 つのディスクリプター として提示する

ユーザーが使いやすくするため、およびここでは論じられていない所定の設計上の 理由のため、マクロ・エディターは、「Field Counts and OIA」ディスクリプター を 1 つのディスクリプターとして提示します (54 ページの図 20 を参照)。しか し、実際には、マクロ・エディターの記述タブ上の「Field Counts and OIA」ディ スクリプターの 3 つの部分はそれぞれ、基礎 XML マクロ言語の別々の独立したデ ィスクリプターに対応します。具体的には、次のとおりです。

- 「フィールド数 (Number of Fields)」の設定は、<numfields> ディスクリプタ ーとして保管される。
- 「入力フィールド数 (Number of Input Fields)」の設定は、<numinputfields> ディスクリプターとして保管される。
- 「OIA が禁止解除になるのを待つ (Wait for OIA to Become Uninhibited)」の 設定は、<oia> ディスクリプターとして保管される。

表 13 は、これらの 3 つのタイプのディスクリプターをリストし、<description> エレメント内に存在できる各ディスクリプターの数を示します。

表 13. 使用できる各ディスクリプター・タイプの数

ディスクリプターのタイプ:	マクロ画面ごと (つまり、 <description> エレメントごと) に使用できるこのタイプの ディスクリプター数:</description>
<oia></oia>	1 (必須)
<numfields></numfields>	1 (オプション)
<numinputfields></numinputfields>	1 (オプション)

上記の表が示すように、これらのディスクリプターのタイプごとに 1 つだけが、 <description> エレメントに存在できます。 <oia> ディスクリプターは必須です が、<numfields> ディスクリプターと <numinputfields> ディスクリプターはオプ ションです。マクロ・エディターがこれらの規則を実行します。

「Field Counts and OIA」ディスクリプターがマクロ・エディターの記述タブに表示されているときに、そのディスクリプターを調べてから、コード・エディターで 調べると、これらの概念を理解できます。 54 ページの図 20 は、マクロ・エディ ターの記述タブ上の「Field Counts and OIA」ディスクリプターを示しています。 「Field Counts and OIA」ディスクリプターの 3 つの部分の設定値は、次のよう に設定されます。

Number of Fields: 80 Number of Input fields: 3 Wait for OIA to Become Uninhibited: true

しかし、コード・エディターを使用して対応する <description> エレメントを調べると、次のように表示されます。

```
<description>
  <oia status="NOTINHIBITED" optional="false" invertmatch="false" />
   <numfields number="80" optional="false" invertmatch="false" />
   <numinputfields number="3" optional="false" invertmatch="false" />
</description>
```

図 24.3 つのディスクリプターを持つ <description> エレメント

上記の XML コード・フラグメントは、<description> エレメントに、3 つの別々 の独立したディスクリプターが含まれていることを示します。各ディスクリプター は、「Field Counts and OIA」ディスクリプターの 3 つの部分のいずれかに対応 しています。

「Field Counts and OIA」ディスクリプターの設定を次のように変更すると仮定します。

Number of Fields: (blank) Number of Input fields: (blank) Wait for OIA to Become Uninhibited: true

最初の 2 つのフィールドをブランクに設定すると、これらの項目がスクリプトに組 み込まれないように、マクロ・エディターに指示します。コード・エディターを使 用して対応する <description> エレメントを再度調べると、次のように表示されま す。

<description>
 <oia status="NOTINHIBITED" optional="false" invertmatch="false" />
</description>

上記の XML コード・フラグメントは、<description> エレメントに、1 つのディ スクリプター、つまり「Field Counts and OIA」ディスクリプターの「OIA が禁 止解除になるのを待つ (Wait for OIA to Become Uninhibited)」設定に対応する <oia> ディスクリプターだけが含まれていることを示しています。

画面認識時の取り扱い

画面認識時に、マクロ・ランタイムが個々のディスクリプターを評価し、ブール結 果を結合するときに、マクロ・ランタイムは、<oia> ディスクリプター、 <numfields> ディスクリプター (存在する場合)、および <numinputfields> ディス クリプター (存在する場合) をそれぞれ、別々の独立したディスクリプターとして、 他のディスクリプターと同じように扱います。

複数のディスクリプターの評価の詳細については、 58 ページの『ディスクリプタ ーの評価』を参照してください。

「OIA が禁止解除になるのを待つ (Wait for OIA to Become Uninhibited)」ディスクリプター (<oia> エレメント)

66 ページの表 14 は、次のことを示しています。

- 「OIA が禁止解除になるのを待つ (Wait for OIA to Become Uninhibited)」リ スト・ボックスの 3 つの可能な設定値
- <oia> エレメントで使用される対応する値
- マクロ・ランタイムが設定をどのように評価するか

表 14. 「OIA が禁止解除になるのを待つ (Wait for OIA to Become Uninhibited)」ディスク リプターの有効な設定値

記述タブ上の設定:	<oia> エレメント内の</oia>	意味:
	status 属性の値:	
true	NOTINHIBITED	セッション・ウィンドウの入 力禁止標識がクリアされる (つまり、入力が禁止されな い)場合、マクロ・ランタイ ムはこのディスクリプターを trueとして評価します。クリ アされない場合、マクロ・ラ ンタイムはこのディスクリプ ターを falseとして評価しま す。
false	DONTCARE	マクロ・ランタイムは、常に このディスクリプターを true として評価します。
<expression></expression>	'NOTINHIBITED'、 'DONTCARE'、またはこれら のストリングのどちらかに評 価される任意の式	マクロ・ランタイムは、式を 評価してから、結果のストリ ングを解釈します。

ほぼすべてのシナリオで、このディスクリプターのデフォルト設定を受け入れることができます。デフォルト設定は、true (記述タブ上) と NOTINHIBITED (マクロ 言語内) です。画面認識時には、次のことが行われます。

- セッション・ウィンドウの入力禁止標識が設定される (つまり、入力が禁止される) 場合、マクロ・ランタイムはこのディスクリプターを false として評価します。
- しかし、入力禁止標識がクリアされる (つまり、入力が禁止されない) 場合、マ クロ・ランタイムはこのディスクリプターを true として評価します。

これらは、予期した結果です。通常、入力禁止標識が設定されている間に、マク ロ・ランタイムがマクロ画面を認識して、アクションの処理をただちに開始するこ とを、ユーザーは希望しません。重要なタイミングの問題がここに含まれていま す。その問題については別のセクションをお読みください(147 ページの『画面の 完了』を参照)。ただし、その問題を解決しても、このディスクリプターは、ほとん どの場合、デフォルト設定 (true)のままにしておく必要があります。

しかし、マクロ・ランタイムに入力禁止状態を無視させるシナリオの場合は、記述 タブでこのディスクリプターを false に設定してください (マクロ言語の相当する 設定は DONTCARE です)。

セッションの指定: このディスクリプターが参照するセッションを指定するには、 「ホスト ID (Host ID)」フィールドを使用します。

- このディスクリプターがマクロを起動するセッションを参照するように指定する には、「ホスト ID (Host ID)」フィールドをブランクのままにする。
- このディスクリプターがその他のセッションを参照するように指定するには、 「ホスト ID (Host ID)」フィールドにセッション参照番号を入力する (196 ペ ージの『ホスト ID の指定』を参照)。

参照されたセッションがアクティブである場合は、このセッションに任意の自動編 集機能を使用できます (197 ページの『異なるセッションでの自動編集機能の使 用』を参照)。

「フィールド数 (Number of Fields)」ディスクリプター (<numfields> エレメント)

「フィールド数 (Number of Fields)」ディスクリプターは、特定数の 3270 (または 5250) フィールドを指定します。「フィールド数 (Number of Fields)」入力フィー ルドには、整数、または整数に評価される任意のエンティティー (例えば、変数、 演算式、または外部 Java メソッドの呼び出し)を使用できます。

画面認識時にマクロ・ランタイムは次のことを行います。

- 1. このディスクリプターを評価し、整数の結果を取得する。
- 2. アプリケーション画面 (現在の状態) 内のフィールド数をカウントする。
- 3. この 2 つの数字を比較する。

2 つの数字が等しい場合、マクロ・ランタイムはこのディスクリプターを true と 評価します。等しくない場合、マクロ・ランタイムはこのディスクリプターを false として評価します。

マクロ・ランタイムがセッション・ウィンドウ内のフィールド数をカウントすると きには、入力フィールドを含めて、すべてのタイプの 3270 (または 5250) フィール ドをカウントします。

このディスクリプターを使用したくない場合は、「フィールド数」入力フィールド をブランクに設定してください。

セッションの指定: このディスクリプターが参照するセッションを指定するには、 「ホスト ID (Host ID)」フィールドを使用します。

- このディスクリプターがマクロを起動するセッションを参照するように指定する には、「ホスト ID (Host ID)」フィールドをブランクのままにする。
- このディスクリプターがその他のセッションを参照するように指定するには、 「ホスト ID (Host ID)」フィールドにセッション参照番号を入力する (196 ペ ージの『ホスト ID の指定』を参照)。

参照されたセッションがアクティブである場合は、このセッションに任意の自動編 集機能を使用できます (197 ページの『異なるセッションでの自動編集機能の使 用』を参照)。

「Number of Input Fields」ディスクリプター (<numinputfields> エレメント)

「入力フィールドの数 (Number of Input Fields)」ディスクリプターは、上記のセ クションで説明した「フィールド数 (Number of Fields)」ディスクリプターと非常 によく似ています。違いは、「入力フィールドの数 (Number of Input Fields)」デ ィスクリプターが 3270 (または 5250) 入力フィールドの数を指定するのに対して、 「フィールド数 (Number of Fields)」ディスクリプターは、すべてのタイプのフィ ールド (入力フィールドを含む) の数を指定することです。 「入力フィールド数 (Number of Input Fields)」フィールドには、整数、または整 数に評価される任意のエンティティー (例えば、変数、演算式、または外部 Java メ ソッドの呼び出し) を使用できます。

画面認識時にマクロ・ランタイムは次のことを行います。

- 1. このディスクリプターを評価し、整数の結果を取得する。
- 2. アプリケーション画面 (現在の状態) 内の入力フィールド数をカウントする。
- 3. この 2 つの数字を比較する。

2 つの数字が等しい場合、マクロ・ランタイムはこのディスクリプターを true と 評価します。等しくない場合、マクロ・ランタイムはこのディスクリプターを false として評価します。

このディスクリプターを使用したくない場合は、「入力フィールド数」フィールド をブランクに設定してください。

セッションの指定: このディスクリプターが参照するセッションを指定するには、 「ホスト ID (Host ID)」フィールドを使用します。

- このディスクリプターがマクロを起動するセッションを参照するように指定する には、「ホスト ID (Host ID)」フィールドをブランクのままにする。
- このディスクリプターがその他のセッションを参照するように指定するには、 「ホスト ID (Host ID)」フィールドにセッション参照番号を入力する (196 ペ ージの『ホスト ID の指定』を参照)。

参照されたセッションがアクティブである場合は、このセッションに任意の自動編 集機能を使用できます (197 ページの『異なるセッションでの自動編集機能の使 用』を参照)。

マクロ開発時のセッション・ウィンドウ内のフィールド数のカウント

「Field Counts and OIA」ディスクリプターを編集しているときに、「フィールド 数」フィールドと「入力フィールド数」フィールドを正しい値に設定したい場合 は、各入力フィールドの隣にある「カレント (Current)」ボタンを使用して、アプリ ケーション画面内のフィールド数をカウントすることができます。

この機能を使用する手順は、次のとおりです。

- セッション・ウィンドウをクリックして、アクティブにする (203 ページの 『セッション・ウィンドウの使用』を参照)。
- 2. セッション・ウィンドウで、作業しているマクロ画面に対応するアプリケーショ ン画面に進む。
- 3. 記述タブで、「Field Counts and OIA」ディスクリプターを選択する。
- 記述タブで、「フィールド数」入力フィールドのすぐ右側にある「カレント (Current)」ボタンをクリックする。マクロ・エディターが、現行アプリケーション画面内のフィールド数をカウントしてから、そのカウントを入力フィールド に表示します。
- 記述タブで、「入力フィールド数」入力フィールドのすぐ右側にある「カレント (Current)」ボタンをクリックする。マクロ・エディターが、現行アプリケーション画面内の入力フィールド数をカウントしてから、そのカウントを入力フィー ルドに表示します。

ストリング・ディスクリプター (<string> エレメント)

ストリング・ディスクリプターは、次の情報を指定します。

- 一連の文字 (ストリング)
- セッション・ウィンドウ上の長方形のテキスト域

マクロ・ランタイムは、長方形のテキスト域全体を検索して、指定されたストリン グを見付けます。マクロ・ランタイムが長方形のテキスト域内でそのストリングを 検出すると、ストリング・ディスクリプターを true と評価します。検出しない場 合、ストリング・ディスクリプターを false と評価します。

長方形域の指定

長方形のテキスト域を定義するには、向かい合った隅の行と列の座標を指定しま す。これらの座標のデフォルト値は (1,1) と (-1,-1) です。これは、セッション・ウ ィンドウのテキスト域全体を示します (-1,-1 などの負の値の意味については、 41 ページの『行または列の負の値の意味』を参照してください)。整数、または整数に 評価される任意のエンティティー (例えば、変数、演算式、または外部 Java メソッ ドの呼び出し)を使用できます。

長方形の領域は、ストリングが入る大きさにするか、ストリングより大きくするこ とができます。例えば、マクロ画面と一致させたいアプリケーション画面の長方形 域 (6,20)、(6,37) に、ストリング 'Terminal and user parameters' があるとしま す。この長方形域は、そのストリングがちょうど入る大きさです。アプリケーショ ン画面のこの位置に常にこのストリングがある場合、正確な長方形域を指定できま す。

しかし、マクロ画面と一致させたいアプリケーション画面上に、同じストリング 'Terminal and user parameters' があるにもかかわらず、アプリケーション画面の どの行にそのストリングが入るか予想できないものとします。この場合、長方形域 (1,1), (-1,-1) を指定できます。これは、マクロ・ランタイムがアプリケーション画面 のすべての行を検索して、その識別ストリングを見付けることを示します。

ストリング値の場合、ストリング、またはストリングに評価される任意のエンティ ティー (例えば、変数、式、または外部 Java メソッドの呼び出し)を使用できま す。このストリングは、選択されたマクロ形式 (基本または拡張)に必要な形式でな ければなりません (204 ページの『ストリングを指定する際のエラー』を参照)。

画面認識時にマクロ・ランタイムは次のことを行います。

- 1. 行と列の値を評価し、値ごとに整数の結果を取得する。
- 2. ストリング値を評価し、ストリングの結果を取得する。
- 3. 行と列の値によって指定されたアプリケーション画面 (現在の状態) 内の長方形 のテキスト・ブロック内の任意の場所で、ストリングを探す。

マクロ・ランタイムが長方形のテキスト・ブロック内でストリングを見付けると、 マクロ・ランタイムは、このディスクリプターを true と評価します。見つからな い場合、マクロ・ランタイムはこのディスクリプターを false として評価します。 マクロ・ランタイムが長方形域を検索する方法 (折り返しオプション) 「折り返し (Wrap)」オプションが false (デフォルト設定) に設定されている場 合、マクロ・ランタイムは、長方形域の各行を別々に検索します。このメソッドが 有効なのは、ストリング全体が 1 つの行に入っている場合です。例えば、ストリン グが Utility Selection Panel であり、長方形域が (1,1)、(24,80) である場合、マ クロ・ランタイムは次のようにこのストリングを検索します。

- 長方形域の最初の行を取得する。ストリングがこの行にあるかどうかを判別します。ない場合は、次の行を検索します。
- 2. 長方形域の 2 番目の行を取得する。ストリングがこの行にあるかどうかを判別 します。ない場合は、次の行を検索します。
- 3. 以降も同様です。

一方、折り返しオプションが true に設定されている場合、マクロ・ランタイム は、次のようにストリングを検索します。

- 1. 長方形域のすべての行を取得し、それらをすべて、順に連結する。
- 2. 連結された行内にストリングがあるかどうかを判別する。

検索しているストリングが、セッション・ウィンドウのある行から次の行に折り返 す可能性がある場合は、折り返しオプションを true に設定してください。このオ プションを、抽出アクションの Unwrap 属性と混同しないください。抽出アクショ ンの Unwrap 属性は、テキスト・ブロックではなく、フィールドに基づきます (90 ページの『「テキストのアンラップ (Unwrap Text)」オプション』を参照)。

次の記述は、折り返しオプションが true に設定されている例を示しています。

図 25 は、アプリケーション画面の行 14 から 18 を示しています。

- 7 Transfer Download ISPF Client/Server or Transfer data set
- 8 Outlist Display, delete, or print held job output
- 9 Commands Create/change an application command table
- * Reserved This option reserved for future expansion

図 25. アプリケーション画面の行 14 から 18

上記の行では、各行の先頭文字はブランク・スペースです。例えば、行 14 では、 先頭の 2 文字は ' 6' です。つまり、ブランクのスペースの後に数表示 6 が続い たものです。このアプリケーション画面で、次の長方形のテキスト・ブロックがあ るかどうか検査するストリング・ディスクリプターをセットアップするものとしま す。

. .

Hardcopy Transfer Outlist Commands Reserved

この複数行のブロックのストリング・ディスクリプターをセットアップする手順 は、次のとおりです。

1. 新しいストリング・ディスクリプターを作成する。

⁶ Hardcopy Initiate hardcopy output

- 2. 「行 (Row)」と「列 (Column)」フィールドに値を設定する。テキスト長方形の 左上隅の行と列の位置は (14, 5) であり、右下隅の行と列の位置は (18, 12) で す。
- 3. ストリング値を設定する。ストリング値は次のとおりです。
 - 'HardcopyTransferOutlist CommandReserved'
- 4. 折り返しオプションを true に設定する。
- 5. これ以外のすべてのオプションをデフォルトに設定したままにする。

上記のステップ 3 では、5 つの行が 1 つのストリングに連結され、充てん文字 (例えば、改行や末尾のスペース) は追加されないことに注意してください。しか し、このストリングには、'Outlist' の後のブランクのスペースが含まれます。こ れは、そのブランクのスペースが、長方形の境界内に含まれるからです。

ストリング・ディスクリプターにおける抽出ストリングの使用: 抽出アクションを 使用して、画面からのテキストをストリング変数に読み取る場合 (87 ページの 『抽出アクション (<extract> エレメント)』を参照)、以降の画面で、ストリング・ ディスクリプターの「ストリング (String)」入力フィールドでそのストリング変数 を使用できます。

例えば、ScreenA で、抽出アクションを使用して、セッション・ウィンドウからの 会社名を、\$strTmp\$ という名前のストリング変数に読み取ることができます。その 後、ScreenB では、ストリング・ディスクリプターで検索されるストリングとして \$strTmp\$ を使用できます。

これを行うことができるのは、折り返しオプションが true に設定されている場合 に、複数行のテキストを抽出するときです。

新規ストリング・ディスクリプターにおける '*' ストリング

新しいストリング・ディスクリプターを作成すると、マクロ・エディターは、初期 のデフォルト値としてストリング '*' を「ストリング (String)」入力フィールドに 入れます。この初期ストリングを消去して、必要なストリングを入力してくださ い。アスタリスク (*) には意味も機能もありません。この初期ストリングは「デフ ォルト・ストリング値」を示し、同じ効果があります。

簡単なパラメーター入力方法

ストリング・ディスクリプターを編集するときに、正しいテキスト長方形とテキス ト・ストリングを指定したい場合、マーキング長方形を使用して、これらの値を設 定できます。この機能を使用する手順は、次のとおりです。

- 1. 記述タブで、編集したいストリング・ディスクリプターを選択する。
- 2. セッション・ウィンドウをクリックして、アクティブにする (203 ページの 『セッション・ウィンドウの使用』を参照)。
- 3. セッション・ウィンドウで、作業しているマクロ画面に対応するアプリケーショ ン画面に進む。
- セッション・ウィンドウで、マーキング長方形を使用して、ストリング・ディス クリプターで使用したいテキスト・ブロックをマークする (203 ページの『マ ーキング長方形の使用』を参照)。
 - セッション・ウィンドウでマーキング長方形を完成すると、マクロ・エディ ターはその長方形の隅の行と列の値を、記述タブ上のストリング・ディスク

リプター・ウィンドウ内の「開始行 (Start Row)、「開始列 (Start Column)」、「終了行 (End Row)」、および「終了列 (End Column)」入力 フィールドに自動的にコピーします。

- 5. セッション・ウィンドウで、「編集 (Edit)」> 「コピー (Copy)」の順にクリッ クする。
- ストリング・ディスクリプター・ウィンドウで、「ストリング (String)」フィー ルドをクリックする。「ストリング」フィールド内のテキスト・カーソルを、ス トリングを追加したい位置に移動してください。
- 7. Ctrl-v をクリックする。
- 複数行のテキストが入っている長方形をセッション・ウィンドウから取り込んだ 場合は、ストリングには追加の編集が必要です。ストリングの編集方法について の下記の説明をお読みください。

Ctrl-v をクリックすると、マクロ・エディターは次のことを行います。

 セッション・ウィンドウ内の長方形ブロックから、「ストリング」フィールドに テキストをコピーする。

しかし、(70 ページの『マクロ・ランタイムが長方形域を検索する方法 (折り返し オプション)』の例のように) 複数行のテキストを取り込んだ場合、貼り付け操作を 行うと、最後の行を除いて、テキストの各行の後に余分のスペースが挿入されま す。「ストリング」フィールドを編集して、これらの余分のスペースを除去してく ださい。 70 ページの『マクロ・ランタイムが長方形域を検索する方法 (折り返し オプション)』の例では、編集前に、「ストリング」フィールドには次のストリング が入っています。

'Hardcopy Transfer Outlist Command Reserved'

このストリングは長過ぎて (4 文字分)、指定されたサイズの長方形 (5 行、8 列) に収まらないこと、および余分なブランクのスペースが各行の後に挿入されている ことに注意してください。このストリングを次の値に編集する必要があります。

'HardcopyTransferOutlist CommandReserved'

同じ <description> エレメント内の複数のストリング・ディスクリ プター

マクロ・エディターでは、同じ <description> エレメント内に複数のストリング・ ディスクリプターを作成できます。信頼性の高い記述を作成するには、必要な数の ストリング・ディスクリプターを使用する必要があります。

同じ <description> エレメントで、同じ長方形のテキスト・ブロックに対して 2 つ の異なるストリングを定義することもできます。これを行うことができるのは、マ クロ画面に対応するアプリケーション画面が、同じロケーションに異なるストリン グを、異なる時間に表示する場合です。しかし、同じ長方形のテキスト・ブロック に 2 つの異なるストリングを定義する場合、両方のストリングが必須でない (両方 ともオプションである) ことを指定するように注意してください。

セッションの指定

このディスクリプターが参照するセッションを指定するには、「ホスト ID (Host ID)」フィールドを使用します。

- このディスクリプターがマクロを起動するセッションを参照するように指定する には、「ホスト ID (Host ID)」フィールドをブランクのままにする。
- このディスクリプターがその他のセッションを参照するように指定するには、 「ホスト ID (Host ID)」フィールドにセッション参照番号を入力する (196 ペ ージの『ホスト ID の指定』を参照)。

参照されたセッションがアクティブである場合は、このセッションに任意の自動編 集機能を使用できます (197 ページの『異なるセッションでの自動編集機能の使 用』を参照)。

カーソル・ディスクリプター (<cursor> エレメント)

カーソル・ディスクリプターは、アプリケーション画面上の行と列の位置 (例え ば、行 10 と列 50)を指定します。行の値と列の値のどちらにも、整数、または整 数に評価される任意のエンティティー (例えば、変数、演算式、または外部 Java メ ソッドの呼び出し)を使用できます。

画面認識時にマクロ・ランタイムは次のことを行います。

- 1. 行の値を評価し、整数の結果を取得する。
- 2. 列の値を評価し、整数の結果を取得する。
- 3. アプリケーション画面 (現在の状態) 内のテキスト・カーソルの行と列の位置を 調べる。
- 4. ディスクリプター内の行と列の位置を、アプリケーション画面のテキスト・カー ソルの行と列の位置と比較する。

2 つの位置が同じである場合、マクロ・ランタイムはこのディスクリプターを true と評価します。 false に評価される場合、マクロ・ランタイムはこのディスクリプ ターを false として評価します。

カーソル・ディスクリプターを編集するときに、正しい行と列の値を指定したい場 合、「カレント (Current)」ボタンを使用できます。「カレント」ボタンは、カーソ ル・ディスクリプター・ウィンドウ内の「行 (Row)」と「列 (Column)」フィール ドのすぐ右側にあります。

この機能を使用する手順は、次のとおりです。

- セッション・ウィンドウをクリックして、アクティブにする (203 ページの 『セッション・ウィンドウの使用』を参照)。
- セッション・ウィンドウで、作業しているマクロ画面に対応するアプリケーション画面に進む。
- 3. セッション・ウィンドウで、マウスまたはキーボードを使用して、必要な位置に テキスト・カーソルを設定する。
- 4. カーソル・ディスクリプター・ウィンドウで、「カレント (Current)」をクリッ クする。

「カレント」をクリックすると、マクロ・エディターは次のことを行います。

 現行のセッション・ウィンドウでテキスト・カーソルの位置を検出し、行と列の 値をその位置に設定する。

セッションの指定

このディスクリプターが参照するセッションを指定するには、「ホスト ID (Host ID)」フィールドを使用します。

- このディスクリプターがマクロを起動するセッションを参照するように指定するには、「ホスト ID (Host ID)」フィールドをブランクのままにする。
- このディスクリプターがその他のセッションを参照するように指定するには、 「ホスト ID (Host ID)」フィールドにセッション参照番号を入力する (196 ペ ージの『ホスト ID の指定』を参照)。

参照されたセッションがアクティブである場合は、このセッションに任意の自動編 集機能を使用できます (197 ページの『異なるセッションでの自動編集機能の使 用』を参照)。

属性ディスクリプター (<attrib> エレメント)

属性ディスクリプターは、3270 または 5250 属性、およびアプリケーション・ウィ ンドウ上の行と列の位置を指定します。

画面認識時に、マクロ・ランタイムは、指定された属性 (例えば、0x3) を、指定された行と列にある実際の属性と比較します。これらの属性が同じである場合、マクロ・ランタイムはこのディスクリプターを true と評価します。異なる場合、マクロ・ランタイムはこのディスクリプターを false として評価します。

属性以外は非常によく似ている 2 つのアプリケーション画面を区別しようとすると きに、このディスクリプターは便利です。

属性の指定

属性を指定する前に、「データ・プレーン (Data Plane)」リスト・ボックスを使用 して、検索しようとしている属性を持つデータ・プレーンを選択してください。 <Expression> を選択する場合は、ランタイム時に、リスト・ボックス内のデータ・ プレーン・ストリングのいずれか (FIELD_PLANE、COLOR_PLANE、または EXFIELD_PLANE) に解決される式 (例えば、\$strDataPlane\$ という名前の変数)を指 定する必要があります。

検索しようとしている属性の、アプリケーション画面上の行と列の位置を指定する には、「行 (Row)」と「列 (Column)」入力フィールドを使用してください。

属性値を指定するには、次の3つの方法のいずれかを使用できます。

- セッション・ウィンドウをクリックし、属性ディスクリプターで指定するような 属性の行と列の位置にテキスト・カーソルを移動してから、マクロ・エディター で「カレント (Current)」をクリックする。
- 「属性を編集 (Edit Attributes)」をクリックし、ポップアップ・ウィンドウ上の 制御機構を使用する。
- 「属性値 (Attribute Value)」入力フィールドに値を入力する (例えば、0x3)。

セッションの指定

このディスクリプターが参照するセッションを指定するには、「ホスト ID (Host ID)」フィールドを使用します。

- このディスクリプターがマクロを起動するセッションを参照するように指定する には、「ホスト ID (Host ID)」フィールドをブランクのままにする。
- このディスクリプターがその他のセッションを参照するように指定するには、 「ホスト ID (Host ID)」フィールドにセッション参照番号を入力する (196 ペ ージの『ホスト ID の指定』を参照)。

参照されたセッションがアクティブである場合は、このセッションに任意の自動編 集機能を使用できます (197 ページの『異なるセッションでの自動編集機能の使 用』を参照)。

条件ディスクリプター (<condition> エレメント)

条件ディスクリプターは、マクロ・ランタイムが画面認識時に評価する条件式 (例 えば、\$intNumVisits\$ == 0)を指定します。条件式の詳細については、 39 ページ の『条件演算子と論理演算子および式』を参照してください。

画面認識時にマクロ・ランタイムは次のことを行います。

• 条件式を評価し、ブールの結果を取得する。

条件式が true に評価される場合、マクロ・ランタイムはこのディスクリプターを true と評価します。 false に評価される場合、マクロ・ランタイムはこのディスク リプターを false として評価します。

条件ディスクリプターは、マクロ・ランタイムが、1 つ以上の変数の値、または Java メソッドの呼び出しの結果に基づいて、次に処理されるマクロ画面を判別でき るようにすることによって、画面認識の柔軟性と能力を向上させます。

カスタム・ディスクリプター (<customreco> エレメント)

カスタム・ディスクリプターでは、カスタム記述コードを呼び出すことができま す。

カスタム・ディスクリプターを使用するには、別個の Host Access Toolkit 製品が 必要です。

カスタム・ディスクリプターを作成するには、コード・エディターを使用して、 <customreco> エレメントをマクロ画面の <description> エレメントに追加しま す。このエレメントの詳細については、 227 ページの『<customreco> エレメン ト』を参照してください。

変数更新アクション (<varupdate> エレメント)

「ディスクリプター」リスト・ボックスに表示される最後のタイプの項目は、変数 更新項目です。これは、ディスクリプターではなく、マクロ言語が <description> エレメント内で許可するアクションです。

<description> エレメント内の変数更新アクションは、<actions> エレメントで実行 するのと同じタイプの操作を実行します。つまり、指定された値を指定された変数 に保管します。 変数更新アクションの作成については、 131 ページの『変数更新アクション (<varupdate> エレメント)』を参照してください。

記述内の変数更新アクションの処理

変数更新アクションが <description> エレメント内にあるときに、マクロ・ランタ イムが1つ以上の変数更新アクションを処理する方法に注意してください。

- 1. マクロ・ランタイムは、すべての変数更新アクションの順番が先頭であるかのよ うに、これらのアクションをただちに実行する。
- 2. その後、マクロ・ランタイムは、記述内の残りの項目 (ディスクリプター)を通 常どおりに評価し、全体的なブール結果を得る。変数更新アクションは、ブール 結果に影響しません。

上記で説明したように、マクロ・ランタイムは、マクロ画面をアプリケーション画 面と一致させる前に、画面認識プロセスを複数回実行する場合があります (48 ペ ージの『評価のやり直し』を参照)。したがって、<description> エレメントに1つ 以上の変数更新アクションが含まれている場合、マクロ・ランタイムは、 <description> エレメントを評価するたびに、変数更新アクションを実行します。

例えば、マクロが再生中であり、画面名 ScreenB が有効な次画面のリスト上にあ り、図 26 に表示されているような <description> エレメントが ScreenB に含まれ ているとします。

```
<description>
   <oia status="NOTINHIBITED" optional="false" invertmatch="false" />
   <varupdate name="$boolUpdate$" value="true" />
  <attrib value="0x4" row="1" col="1" plane="COLOR_PLANE" optional="false"
               invertmatch="false" />
```

</description>

図 26. ScreenB の <description> エレメント

マクロ・ランタイムが ScreenB を現行のアプリケーション画面と突き合わせようと するたびに、次のことを行います。

- 1. マクロ・ランタイムは、<varupdate> アクションを参照し、それを実行して、 値 true を \$boolUpdate\$ に保管する。
- 2. マクロ・ランタイムは、<oia> ディスクリプターと <attrib> ディスクリプター を評価し、<description> エレメント全体のブール結果を得る。

uselogic 属性を使用した変数更新

<description> エレメント内の変数更新アクションを、先頭以外の順序でマクロ・ラ ンタイムに実行させたい場合は、デフォルト結合規則ではなく、<description> エレ メントの uselogic 属性を使用して、実行の順序を変更できます (61 ページの 『uselogic 属性』を参照)。

uselogic 属性で変数更新アクションが使用される場合は、次のようになります。

- マクロ・ランタイムは、uselogic 属性で指定されるのと同じ順序で変数更新アク ションを実行する。
- マクロ・ランタイムは、常に変数更新アクションを true に評価する。

第8章 マクロ・アクション

概要

機能別のアクション

次に、すべてのアクションのリストを機能別にグループ分けして示します。

- ユーザーとの対話
 - ボックス選択
 - 入力 (キー・ストローク、およびクリップボードへのコピーなど、キーによっ てアクティブになる機能)
 - メッセージ
 - マウス・クリック
 - プロンプト
- アプリケーションからのデータの取得とアプリケーションへのデータの送信
 - 抽出
 - FileUpload
 - SQLQuery
 - 転送 (ホストとクライアント間でのアップロードまたはダウンロードのファイ ル転送)
 - 変数更新
- 待ち
 - 通信待機
 - 休止
- プログラミング
 - 条件
 - 実行 (Java メソッドの呼び出し)
 - PlayMacro (別のマクロへのチェーン)
 - プログラム実行 (クライアント・オペレーティング・システム上でプログラム を起動する)
 - 変数更新
- ・ システム
 - 印刷抽出
 - 印刷開始
 - 印刷終了
 - プログラム実行 (クライアント・オペレーティング・システム上でプログラム を起動する)
- デバッグ
 - トレース

ランタイムのコンテキスト

43 ページの『第6章マクロ・ランタイムによるマクロ画面の処理方法』で説明 したとおり、マクロ・ランタイムが新しい現行マクロ画面を選択した場合、マク ロ・ランタイムは、そのマクロ画面の <actions> エレメントにあるアクションの実 行を即時に開始します。

すべてのアクションを実行した後、マクロ・ランタイムは、次に処理するマクロ画 面を判別する次のステップにすぐ進みます。

マクロ画面のコンテキスト

単一のマクロ画面内では、マクロ・ランタイムは <actions> エレメント内での出現 順にアクションを実行します。この順序は、「アクション (Action)」リスト・ボッ クスで設定したアクションの順序と同じです。

マクロ画面のアクションの作成は必須ではありません。 <actions> エレメントが含まれていない場合、または <actions> エレメントが空である場合、マクロ・ランタイムはマクロ画面処理の次のセクション (次に処理するマクロ画面の選択) に直行します。

アクションのパラメーターの指定

一般に、アクションのパラメーターを指定する際には、特定の標準データ・タイプ の即時値を受け入れるコンテキストはすべて、同じデータ型のエンティティーをど れでも受け入れることに注意してください。例えば、ある入力フィールドがストリ ング値を受け入れる場合、そのフィールドは、ストリングに評価される式、ストリ ングに変換される値、ストリング変数、またはストリングを戻すインポート済みメ ソッドの呼び出しも受け入れます (41 ページの『等価』を参照)。

ただし、変数を使用できないフィールドがいくつかあります (161 ページの『変数 が保持する値の使用』を参照)。

「アクション (Actions)」タブの概要

「アクション (Actions)」タブの例

マクロ・エディターの「画面 (Screens)」タブにある「アクション (Actions)」タブ を使用して、アクションを作成および編集できます。「アクション (Actions)」タブ でアクションを作成すると、マクロ・エディターは、現在選択されているアクショ ンの <actions> エレメントに新規アクションを挿入します。 79 ページの図 27 に、「アクション (Actions)」タブの例を示します。

ホスト・アクセス・マクロ・エディター		
マクロ 画面 リンク 変数		
画面名 Screen1 画面を削除		
一般 記述 アクション		
アクション 定義されたアクションなし 削除 順序変更 定義されたアクションなし (新規の入力アクション> (新規のプロンプト・アクション> <新規のガロンプト・アクション> (新規のパッセージ・アクション> <新規の休止アクション> (新規の転送アクション> <新規の転送アクション> (新規の通信待機アクション>		
ストリング		
アクション・キー [altview] [attn]		
ホスト・アクション・キーの変換 false 📃		
カーソルを入力の最後に移動 false 🔍		
□ パスワード		
ホスト ID		
保管して終了 別名保管… 保管 キャンセル コード・エディター… インポート… エクスポート… ヘルプ		
この画面が認識されると、画面にテキストを入力する		

図 27. 「アクション (Actions)」タブ

上の図では、マクロ・エディターの「画面 (Screens)」タブが選択されています。現 在選択されている画面の名前 (Screen1) が、「画面 (Screens)」タブの「画面名 (Screen Name)」フィールドに表示されています。「画面名 (Screen Name)」フィ ールドの下には、「一般 (General)」、「記述 (Description)」、および「アクショ ン (Actions)」のサブタブがあります。「アクション (Actions)」タブが選択されて います。

上の図の「アクション (Actions)」タブに注目すると、「記述 (Description)」タブ と同様、上部領域と下部領域があることが分かります。

上部領域には、単一のアクション・エレメントを全体と見なして作用するコントロ ールがあります。具体的には、「アクション (Actions)」タブの左上隅にある「アク ション (Action)」リスト・ボックスに、現在選択されているアクションの名前が表 示されます。上の図では、まだアクションが作成されていないので、現在選択され ているアクションはありません。

「アクション (Actions)」タブの下部領域には、現在選択されているアクション (あ れば)の内容が表示されます。現在選択されているディスクリプターが入力アクシ ョンの場合、「アクション (Actions)」タブの下部領域にはそのタイプのアクション に該当する内容が表示されます。抽出アクションなど、別のタイプのアクションを ユーザーが作成または選択した場合、下部領域には抽出アクションに該当する内容 が表示されます。

新規アクションの作成

79 ページの図 27 の「アクション (Actions)」リスト・ボックスにもう一度注目す ると、このリスト・ボックスにはまだアクションが入っていないことが分かりま す。不等号括弧で囲まれ、「新規 (new)」という語で始まる選択項目は、すべて新 規アクションの作成用です。 79 ページの図 27でも分かるように、「アクション (Actions)」リスト・ボックスの表示可能部分は、リスト全体を一度に表示するには 高さが不足しています。リストの全体は次のとおりです。

<新規の入力アクション> (<new input action>) <新規の抽出アクション> (<new extract action>) <新規のプロンプト・アクション> (<new prompt action>) <新規のメッセージ・アクション> (<new message action>) <新規の休止アクション> (<new pause action>) <新規の転送アクション> (<new xfer action>) <新規の通信待機アクション> (<new comm wait action>) <新規のトレース・アクション> (<new trace action>) <新規のマウス・クリック・アクション> (<new mouse click action>) <新規のボックス選択アクション> (<new box select action>) <新規のプログラム実行> (<new run program>) <新規の変数更新アクション> (<new variable update action>) <新規のマクロ再生アクション> (<new play macro action>) <新規の実行アクション> (<new perform action>) <新規の条件アクション> (<new conditional action>) <新規の印刷開始アクション> (<new print start action>) <新規の印刷抽出アクション> (<new print extract action>) <新規の印刷終了アクション> (<new print end action>) <new sql query action> <new file upload action>

図 28. 作成されたアクションがない状態での「アクション (Actions)」リスト・ボックスのリ ストの内容

例えば、「<新規の入力アクション> (<new input action>)」をクリックすると、マ クロ・オブジェクトは新規の入力アクションを作成し、リストの一番上に配置しま す。「アクション (Actions)」タブの下部領域では、入力アクションに属する各種フ ィールド (入力キー・シーケンスなど) に入力できます。「アクション (Actions)」 リスト・ボックスの選択領域に新規の入力項目が表示され、リスト・ボックスのリ スト部分は次のようになります。

Input action1(0,0) <新規の入力アクション> (<new input action>) <新規の抽出アクション> (<new extract action>) <新規のプロンプト・アクション> (<new prompt action>) <新規のメッセージ・アクション> (<new message action>) <新規の休止アクション> (<new pause action>) <新規の転送アクション> (<new xfer action>) <新規の通信待機アクション> (<new comm wait action>) <新規のトレース・アクション> (<new trace action>) <新規のマウス・クリック・アクション> (<new mouse click action>) <新規のボックス選択アクション> (<new box select action>) <新規のプログラム実行> (<new run program>) <新規の変数更新アクション> (<new variable update action>) <新規のマクロ再生アクション> (<new play macro action>) <新規の実行アクション> (<new perform action>) <新規の条件アクション> (<new conditional action>) <新規の印刷開始アクション> (<new print start action>) <新規の印刷抽出アクション> (<new print extract action>) <新規の印刷終了アクション> (<new print end action>) <new sql query action> <new file upload action>

図 29. 実際のアクションが 1 つある状態での「アクション (Actions)」リスト・ボックスの リストの内容

マクロ・ランタイムがこのマクロ画面を処理する際には、「アクション (Actions)」 リスト・ボックスのリスト順と同じ順序でアクションが実行されます。実際のアク ションの順序を変更するには、「アクション (Actions)」リスト・ボックスの右側に ある「順序変更 (Change Order)」ボタンをクリックします。

アクション

ボックス選択アクション (<boxselection> エレメント)

ボックス選択アクションは、セッション・ウィンドウにマーキング長方形を描画し ます。このアクションは、実際のユーザーがセッション・ウィンドウをクリック し、マウス・ボタン 1 を押したまま、マウスをドラッグしてマーキング長方形を作 成するアクションをシミュレートします。

行と列の値の指定

「アクション (Actions)」タブの下部領域で、作成または編集するマーキング長方形 の四隅の位置を行と列によって指定します。また、ボックス選択アクションが「ア クション (Actions)」リスト・ボックスで現在選択されているアクションならば、セ ッション・ウィンドウの上でマウスをクリックしてドラッグし、目的のマーキング 長方形を作成できます。セッション・ウィンドウ上でマーキング長方形を完成させ ると、マクロ・エディターは、マーキング長方形の隅の行と列の値をボックス選択 アクションの「行 (Row)」と「列 (Column)」の入力フィールドに書き込みます。

「行 (下隅) (Row (Bottom Corner))」入力フィールドと「列 (下隅) (Column (Bottom Corner))」入力フィールドに負の数値 (-1 など) を入力すると、セッショ ン・ウィンドウの最後の行と最後の行からの相対オフセットを指定できます。例え ば、セッション・ウィンドウが 24 x 80 で、隅を (1, 1) と (-4, -10) に指定した場 合、座標 (1, 1) と (21, 71) のマーキング長方形がマクロ・ランタイムによって描 画されます (41 ページの『行または列の負の値の意味』を参照)。

マーキング長方形の消去

マーキング長方形を描画してから、セッション・ウィンドウ内でマーキング長方形 の境界の外側をクリックすると、セッション・ウィンドウからマーキング長方形が 消去されます。ただし、マクロ・ランタイムがマクロを実行しているときに、マウ ス・クリック・アクションは既存のマーキング長方形を消去しません。

マクロ内でマーキング長方形を消去するには、次の 2 つの方法を選択できます。

- 新しいマーキング長方形を別の位置に描画する。新しいマーキング長方形を描画 すると、以前のものが消去されます。
- コード・エディターを使用して、<boxselection> エレメントの type 属性を DESELECT に設定する。 DESELECT オプションの場合、マクロ・ランタイム は行と列の座標を無視し、既存のマーキング長方形があれば単にそれを消去しま す。

セッションの指定

「ホスト ID (Host ID)」フィールドを使用して、このアクションを実行するセッションを指定します。

- マクロを起動するセッションでこのアクションを実行する場合は、「ホスト ID (Host ID)」フィールドをブランクのままにする。
- 別のセッションでこのアクションを実行する場合は、「ホスト ID (Host ID)」 フィールドにセッション参照を入力する(196 ページの『ホスト ID の指定』を 参照)。

参照されたセッションがアクティブである場合は、このセッションに任意の自動編 集機能を使用できます (197 ページの『異なるセッションでの自動編集機能の使 用』を参照)。

例

105 ページの『コピーと貼り付けの例』を参照してください。

通信待機アクション (<commwait> エレメント)

通信待機アクションは、セッションの通信状況がアクションに指定した状態に変わ るまで待ちます。例えば、セッションが完全に接続状態になるまで待つ通信待機ア クションを作成できます。

アクションの動作

マクロ・ランタイムが通信待機アクションの実行を開始すると、ランタイムは通信 待機アクションに指定された通信状況を注視し、セッションの実際の通信状況と比 較します。2 つの状況が一致した場合、マクロ・ランタイムは通信待機アクション が完了したと判断し、マクロ・ランタイムは次のアクションの実行に進みます。

ただし、2 つの状況が一致しない場合、マクロ・ランタイムは他の処理を行わず、 通信待機アクションに指定された通信状況がセッション内で実際に発生するまで単 に待ちます。 タイムアウト値の有効期間が切れるとマクロ・ランタイムが通信待機アクションを 終了するように、通信待機アクションにミリ秒単位のタイムアウト値を指定できま す。つまり、マクロ・ランタイムが待っている通信状況にならなくても、タイムア ウト値の有効期限が切れるとマクロ・ランタイムはアクションを終了します。

通信待機アクションの後、抽出アクションなど他のアクションを使用してアプリケ ーション画面の特性を検査し、セッションが実際に目的の通信状況になったのか、 それとも通信待機アクションがタイムアウトのために終了したのかを判別する必要 が生じることがあります。

永続的な通信状況の指定

通常、セッションの接続または切断時には、通信状況はいくつかの状態を素早く通 過してから (例えば、保留アクティブ、アクティブ、レディーの順に)、安定した状 態を一定期間保つ特定の状態 (ワークステーション ID レディーなど) に達します。 ほとんどの場合、その永続的な終了状態を通信待機アクションに指定する必要があ ります。永続状態のリストについては、次のセクションを参照してください。

(代わりに保留アクティブなどの過渡的な状態を指定した場合、マクロ・ランタイム が通信待機アクションを実行する機会を得る前に、セッションがその状態を通過し て次の状態に進む可能性があります。このため、マクロ・ランタイムが通信待機ア クションを実行すると、すでに発生した状態をいつまでも待つことになります。)

通信状態

「通信状況 (Connection Status)」リスト・ボックスにリストした状況のいずれかを 指定できます。表 15 に、それぞれの状態の名前と意味をリストします。

通信状態:	意味:
通信開始 (Connection Initiated)	初期状態。通信の開始が実行された。
接続保留アクティブ (Connection Pending	ソケット接続の要求。
Active)	
接続アクティブ (Connection Active)	ソケット接続済み。ホストとの接続。
接続レディー (Connection Ready)	Telnet 折衝が開始された。
接続装置名レディー (Connection Device	装置名が折衝された。
Name Ready)	
接続ワークステーション ID レディー	ワークステーション ID が折衝された。
(Connection Workstation ID Ready)	
接続保留非アクティブ (Connection Pending	通信の停止が実行された。
Inactive)	
接続非アクティブ (Connection Inactive)	ソケットがクローズされた。ホストとの接続
	なし。

表 15. 通信状態

安定状態 (つまり、通常は数秒間よりも長く持続する状態) には次のものがありま す。

- 接続非アクティブ セッションが完全に切断された状態。
- 接続ワークステーション ID レディー セッションが完全に接続された状態。

「接続状況 (Connection Status)」リスト・ボックスで「<式> (<Expression>)」を 選択した場合は、 <commwait> エレメントの value 属性に指定されていることを マクロ・ランタイムが予期する、いずれかのキーワードに解決される式を指定する 必要があります (223 ページの『<commwait> エレメント』を参照)。例えば、 CONNECTION READY に解決される変数 \$strCommState\$ を指定できます。

セッションの指定

「ホスト ID (Host ID)」フィールドを使用して、このアクションを実行するセッションを指定します。

- マクロを起動するセッションでこのアクションを実行する場合は、「ホスト ID (Host ID)」フィールドをブランクのままにする。
- 別のセッションでこのアクションを実行する場合は、「ホスト ID (Host ID)」 フィールドにセッション参照を入力する (196 ページの『ホスト ID の指定』を 参照)。

参照されたセッションがアクティブである場合は、このセッションに任意の自動編 集機能を使用できます (197 ページの『異なるセッションでの自動編集機能の使 用』を参照)。

例

<commwait value="CONNECTION_READY" timeout="10000" />

図 30. 通信待機アクションの例

条件アクション (<if> エレメントおよび <else> エレメント)

条件アクションには次の項目があります。

- マクロ・ランタイムが真または偽として評価する条件式。
- 条件が真として評価された場合にマクロ・ランタイムが実行するアクションのリスト。(オプショナル)
- 条件が偽として評価された場合にマクロ・ランタイムが実行するアクションのリスト。(オプショナル)

条件アクションには、if 文または if-else 文の機能があります。

条件の指定

「条件 (Condition)」フィールドに、マクロ・ランタイムが評価する条件式を入力す る必要があります。条件式には論理演算子と条件演算子を指定でき、また演算式、 即時値、変数、および Java メソッドの呼び出しを含む項を指定できます (39 ペー ジの『条件演算子と論理演算子および式』を参照)。

条件が真 (<if> エレメント)

条件が真として評価された場合に実行するアクションを指定するには、「条件が真 (Condition is True)」タブを使用します。

「条件が真 (Condition is True)」タブには、「アクション (Actions)」タブのコン トロールとほとんど同一のコントロールがあります。具体的には、次のとおりで す。

- 「条件が真 (Condition is True)」タブの「アクション (Action)」リスト・ボッ クスでは、「アクション (Actions)」タブの「アクション (Action)」リスト・ボ ックスと同じ方法でアクションを作成および編集できます。
- 「条件が真 (Condition is True)」タブの「削除 (Delete)」ボタンと「順序変更 (Change Order)」ボタンを使用すると、「アクション (Actions)」タブの「削除 (Delete)」ボタンと「順序変更 (Change Order)」ボタンと同じ方法で、アクショ ンを削除または再配列できます。
- 「条件が真 (Condition is True)」タブの下部領域では、「アクション (Actions)」タブの下部領域と同じ方法で、現在選択されているアクションの値を 編集できます。

「条件が真 (Condition is True)」タブにあるこれらのコントロールを使用して、条件が真の場合にマクロ・ランタイムが実行するアクションを作成および編集します。

例えば、「条件 (Condition)」フィールドを \$boolResult\$ のような変数の名前に設 定できます。この変数には、前の操作が結果を保管しています。 \$boolResult\$ の値 が真ならば、状況メッセージをユーザーに表示するメッセージ・アクションを実行 したいとします。この場合は、「条件が真 (Condition is True)」タブでメッセー ジ・アクションを作成し、表示したい状況メッセージを指定できます。

マクロの再生時に、マクロは条件 \$boolResult\$ を評価します。値が真ならば、マク ロ・ランタイムは「条件が真 (Condition is True)」タブで定義したメッセージ・ア クションを実行します。値が偽ならば、マクロ・ランタイムはメッセージ・アクシ ョンをスキップし、また「条件が真 (Condition is True)」タブで定義した他のすべ てのメッセージ・アクション (あれば) をスキップします。

条件が偽 (<else> エレメント)

条件が偽として評価された場合に実行するアクションを指定するには、「条件が偽 (Condition is False)」タブを使用します。

「条件が真 (Condition is True)」タブと同様、「条件が偽 (Condition is False)」 タブには、「アクション (Actions)」タブのコントロールとほとんど同一のコントロ ールがあります。「条件が偽 (Condition is False)」タブにあるこれらのコントロー ルを使用して、条件が偽の場合にマクロ・ランタイムが実行するアクションを作成 および編集します。

条件アクションの中で使用できない条件アクション

マクロ・エディターの「条件が真 (Condition is True)」タブまたは「条件が偽 (Condition is False)」タブ内では、条件アクションを作成できません。このため、 別の if 文の中にネストされた if 文や、else 文の中にネストされた if 文に相当す るものは使用できません。コード・エディターにも同じ規則があります。

例

次のコード・フラグメントは、ユーザーに入力を促します。入力ストリングがスト リング true ならば、コード・フラグメントはメッセージ・ウィンドウにメッセー ジ "You typed TRUE" を表示します。入力ストリングが他のストリングならば、コ ード・フラグメントはメッセージ・ウィンドウにメッセージ "You typed FALSE" を表示します。この例では、プロンプト・アクション、条件アクション、およびメ ッセージ・アクションを使用します。

このコード・フラグメントを本書からシステム・クリップボードにコピーし、シス テム・クリップボードからコード・エディターにコピーできます。このコードはフ ラグメントなので、マクロ画面内で既存のマクロ・スクリプトの中にコードをコピ ーする必要があります。また、\$strData\$ という名前のストリング変数を作成する必 要があります。変数を作成するには、<HAScript> 開始タグの後、最初の <screen> エレメントの前に次の行を追加します。

<vars>

<create name="\$strData\$" type="string" value="" />
</vars>

スクリプトをマクロ・エディターに保管した後、マクロ・エディターまたはコー ド・エディターを使用してスクリプトを編集できます。

この例については、次の点に注意してください。

- この例は、<actions> エレメントを含むコード・フラグメントと、このエレメン ト内にあるアクションで構成されます。
- 最初のアクションは、メッセージ・ウィンドウを表示し、ユーザーの入力を変数 \$strData\$ にコピーするプロンプト・アクションです。入力は、セッション・ウィンドウの入力フィールドには書き込まれません。
- 条件アクション (<if>エレメント)の最初の部分には、条件があります。条件は 単に \$strData\$ です。
- \$strData\$ はブール・コンテキストのストリング変数なので、マクロ・ランタイムはストリングのブール値への変換を試行します(40ページの『自動データ型変換』を参照)。ユーザーの入力がストリング 'true'(大文字、小文字、または大/小文字混合)ならば、変換は正常に行われ、条件にブール値 true が入ります。ユーザーの入力が他のストリングならば、変換は失敗し、条件にはブール値false が入ります。
- 条件が真ならば、マクロ・ランタイムは <if>エレメント内のアクションを実行 します。これは、メッセージ You typed TRUE を表示するメッセージ・アクシ ョンです。その後、条件が真の場合に実行するアクションはすべて行われたので (ここではアクションは 1 つだけ)、マクロ・ランタイムは <else> アクションを スキップし、通常どおり継続します。
- これに対し、条件が偽ならば、マクロ・ランタイムは <if>エレメントにあるア クションをスキップし、<else>エレメント内のアクションの実行を開始します。 このエレメントには、メッセージ You typed FALSE を表示するメッセージ・ア クションが 1 つあります。<else>アクション内のアクションをすべて実行した 後、マクロ・ランタイムは通常どおり継続します。

```
<actions>
<prompt name="'Type true or false'" description="" row="0" col="0"
len="80" default="" clearfield="false" encrypted="false"
movecursor="true" xlatehostkeys="true" assigntovar="$strData$"
varupdateonly="true" />
<if condition="$strData$" >
<message title="" value="'You typed TRUE'" />
</if>
<else>
<message title="" value="'You typed FALSE'" />
</else>
</actions>
```

図 31. 条件アクションを示すサンプル・コード・フラグメント

抽出アクション (<extract> エレメント)

抽出アクションは、セッション・ウィンドウからテキストを抽出し、変数にテキストを格納します。このアクションは非常に有用で、アプリケーション・データを読み取るためにマクロ・オブジェクトが提供している基本方式です (ツールキットの プログラミング API を使用する場合を除けば)。

IBM Host Access Toolkit 製品を使用している場合は、抽出アクションを使用して、使用可能なデータ・プレーンのいずれかからデータを読み取ることができます。詳しくは、 92 ページの『Toolkit を使用したデータ・プレーンからのデータのキャプチャー』を参照してください。

テキストのキャプチャー

抽出アクションの最も一般的な用途は、セッション・ウィンドウに表示されている テキストのキャプチャーです。この操作のために IBM Host Access Toolkit は必要 ありません。

次に、行う手順の概要を示します。その後のサブセクションで、各ステップについ て詳しく説明します。

- 1. 「連続抽出 (Continuous Extract)」オプションを設定する (必要な場合)。
- 2. キャプチャーするセッション・ウィンドウの領域を指定する。
- 3. 抽出名を指定する。
- 4. データ・プレーンとして TEXT_PLANE を指定する。
- 5. テキストを格納する変数を指定する。

「連続抽出 (Continuous Extract)」オプションを設定する: テキストの長方形ブロ ックをキャプチャーする場合は、「連続抽出 (Continuous Extract)」オプションを false に設定します (これがデフォルト値です)。詳しくは、 89 ページの『セッショ ン・ウィンドウの長方形領域のキャプチャー』を参照してください。

これに対し、行から行に折り返すテキストの連続シーケンスをキャプチャーする場 合は、「連続抽出 (Continuous Extract)」オプションを true に設定します。詳しく は、 90 ページの『セッション・ウィンドウからのテキスト・シーケンスのキャプ チャー』を参照してください。 セッション・ウィンドウの領域を指定する: キャプチャーするセッション・ウィン ドウの領域を指定するには、マーキング長方形を使用して行と列の座標を取り込む ことができます。また別の方法として、「抽出アクション (Extract action)」ウィン ドウの「行 (Row)」フィールドと「列 (Column)」フィールドに、テキスト域の行 と列の座標を入力することもできます。

どちらの方式を使用する場合も、マクロ・ランタイムは、「連続抽出 (Continuous Extract)」オプションが false または true のいずれに設定されているかによって、 値を異なった方法で解釈します (87 ページの『「連続抽出 (Continuous Extract)」オプションを設定する』を参照)。

マーキング長方形を使用する場合(203 ページの『マーキング長方形の使用』を参照)、マクロ・ランタイムは、マーキング長方形の左上隅の行と列の座標を1 組目の行と列の値(「抽出アクション (Extract action)」ウィンドウの「上隅 (Top Corner)」)に書き込み、右下隅の行と列の座標を2 組目の行と列の値(「下隅 (Bottom Corner)」)に書き込みます。

行と列の値をユーザー自身が入力する場合は、1 組目の行と列の座標を 1 組目の行 と列の値(「抽出アクション (Extract action)」ウィンドウの「上隅 (Top Corner)」)に入力し、2 組目の座標のセットを 2 組目の行と列の値(「下隅 (Bottom Corner)」)に入力します。必要な座標を判別するための補助として、セッ ション・ウィンドウでテキスト・カーソルを使用できます(203 ページの『セッシ ョン・ウィンドウのテキスト・カーソルの使用』を参照)。

「行 (下隅) (Row (Bottom Corner))」入力フィールドに -1 を入力すると、セッシ ョン・ウィンドウのデータ域の最終行を示すことができます。この機能は、ユーザ ーが高さの異なる (25、43、50 など) セッション・ウィンドウを使用していて、最 終行までデータをキャプチャーしたい場合に便利です。同様に、「列 (下隅) (Column (Bottom Corner))」入力フィールドに -1 を入力すると、セッション・ウ ィンドウのデータの最終列を示すことができます (41 ページの『行または列の負 の値の意味』を参照)。

抽出名を指定する: 'Extract1' などの抽出名を指定できます。ただし、IBM Host Access Toolkit 製品を使用していないかぎり、いかなる目的でもこの名前を使用することはできません。

データ・プレーンとして TEXT_PLANE を指定する: 「データ・プレーン (Data Plane)」リスト・ボックスの「TEXT_PLANE」をクリックします。これはデフォルトです。

テキストを格納する変数を指定する: チェック・ボックス「テキスト・プレーンを 変数に割り当てる (Assign Text Plane to a Variable)」を設定し、テキストを格納 する変数の名前を入力します。 IBM Host Access Toolkit 製品を使用していない場 合は、この方式を使用してキャプチャーしたテキストを格納する必要があります。

テキストはストリングとして戻されます。ほとんどの場合、マクロ内の他のアクションがストリングを処理できるように、ストリングはストリング変数に格納する必要があります。

ただし、他の標準データ型 (boolean、integer、double) を指定した場合、マクロ・ ランタイムは (可能であれば) ストリングを変数のフォーマットに変換します。例え ば、画面のテキストが 1024 で、変数が整変数の場合、マクロ・ランタイムはスト リング 1024 を整数の 1024 に変換し、整変数に値を格納します。フォーマットが 無効でストリングを変数のデータ型に変換できない場合、マクロ・ランタイムはラ ンタイム・エラーを出してマクロを終了します。データ変換について詳しくは、 40 ページの『自動データ型変換』を参照してください。

ヌルなどの表示不可文字の処理

TEXT_PLANE からキャプチャーしたテキストには、ヌル (0x00) などの表示不可文 字は含まれていません。ブランク・スペース文字が入って表示されている表示画面 の文字セルは、ブランク・スペース文字としてキャプチャーされます。

セッション・ウィンドウの長方形領域のキャプチャー

「連続抽出 (Continuous Extract)」オプションが false の場合 (これがデフォルト 値です)、マクロ・ランタイムは、2 組の列と行の値をテキストの長方形ブロックの 左上隅と右下隅 (隅を範囲に含む) として扱います。長方形ブロックは、最小で 1 文字、最大でアプリケーション・ウィンドウ全体にすることができます。

マクロ・ランタイムは、次の処理を行います。

- 結果ストリングを空ストリングに初期化する。
- テキストの長方形ブロックを1行ずつ読み取り、それぞれの行を結果ストリングに連結する。
- 結果ストリングを指定された変数に格納する。

ー例として、セッション・ウィンドウの行 16、17、18 の先頭 40 文字が次のとお りであるとします。

.8..Outlist....Display, delete, or prin .9..Commands....Create/change an applica .10.Reserved....This option reserved for

さらに、マクロ・ランタイムが次の設定値を指定して抽出アクションを実行すると します。

- 「連続抽出 (Continuous Extract)」は false。
- 行と列の組は (16, 5) (Outlist の 'O') と (18, 12) ('Reserved' の 'd')。
- 抽出名は 'Extract1'。
- データ・プレーンは TEXT_PLANE。
- ストリング変数 \$strTmp\$ が結果ストリングの格納先の変数。

「連続抽出 (Continuous Extract)」オプションが false なので、マクロ・ランタイ ムは行と列の組をテキストのマーキング長方形ブロックとして扱い、左上隅は行 16、列 5、右下隅は行 18、列 12 とします。

マクロ・ランタイムは、結果ストリングを空ストリングに初期化します。その後、 マクロ・ランタイムはテキストの長方形ブロックを一度に 1 行ずつ読み取り ('Outlist.'、'Commands'、'Reserved')、それぞれの行を結果ストリングに連結しま す。最後に、マクロ・ランタイムは結果ストリング全体を結果変数 \$strTmp\$ に格 納します。これにより、変数 \$strTmp\$ には次のストリングが格納されます。 'Outlist.CommandsReserved' セッション・ウィンドウからのテキスト・シーケンスのキャプチャー

「連続抽出 (Continuous Extract)」オプションが true の場合、マクロ・ランタイム は、2 組の行と列の値を、テキストの連続シーケンスの開始位置と終了位置 (これ らの位置を範囲に含む) として扱います。テキストは、開始位置から終了位置に至 るまでに必要ならば行間で折り返します。テキストのシーケンスは、最小で 1 文 字、最大でアプリケーション・ウィンドウ全体にすることができます。

マクロ・ランタイムは、次の処理を行います。

- 結果ストリングを空ストリングに初期化する。
- テキストの長方形ブロックを最初から最後まで読み取り、必要な場合は行の最後 から次の行の最初まで折り返す。
- 結果ストリングを指定された変数に格納する。

例えば、セッション・ウィンドウの行 21 と 22 に次のテキストがあるとします (各行は 80 文字)。

.....Enter / on the data set list command field for the command prompt pop-up or ISPF line command.....

さらに、マクロ・ランタイムが次の設定値を指定して抽出アクションを実行すると します。

- ・ 「連続抽出 (Continuous Extract)」は true。
- 行と列の組は (21, 9) ('Enter' の 'E') と (22, 20) ('command' の 'd')。
- 抽出名は 'Extract1'。
- データ・プレーンは TEXT_PLANE。
- ストリング変数 \$strTmp\$ が結果ストリングの格納先の変数。

「連続抽出 (Continuous Extract)」オプションが true なので、マクロ・ランタイム は行と列の組をテキスト・シーケンスの開始と終了のマーキングとして扱い、開始 位置は (21, 9)、終了位置は (22, 20) とします。

マクロ・ランタイムは、結果ストリングを空ストリングに初期化します。その後マ クロ・ランタイムは、テキスト・シーケンスを最初から最後まで読み取り、行 21 の最後の文字から行 22 の最初の文字に折り返します。最後に、マクロ・ランタイ ムは結果ストリング全体を結果変数 \$strTmp\$ に格納します。変数 \$strTmp\$ に は、次に示す 92 文字のストリングが格納されます (次に示すテキストは、本書の ページに収めるために行末ハイフン付けが行われていますが、実際にはハイフンな しの 1 つのストリングを表します)。

'Enter / on the data set list command field for the command prompt pop-up or ISPF line command'

これに対し、この例で「連続抽出 (Continuous Extract)」オプションを false に設 定した場合、\$strTmp\$ には 24 文字のストリング 'Enter / on tline command' が 格納されます。

「テキストのアンラップ (Unwrap Text)」オプション

このオプションは、元は IBM Host Access Toolkit 製品と組み合わせて、「連続抽 出 (Continuous Extract)」オプションが false に設定されている場合にのみ使用す るものです。ただし、ツールキットがなくてもこのオプションを使用でき、「連続 抽出 (Continuous Extract)」オプションが false または true のどちらに設定され ていても使用できます。

「テキストのアンラップ (Unwrap Text)」を true に設定すると、マクロ・ランタ イムは、「抽出 (Extract)」ウィンドウの行と列の組だけでなくセッション・ウィン ドウのフィールド境界も使用して、収集するデータを判別します。マクロ・ランタ イムは、ストリングの配列 (ツールキットを使用している場合)、または単一の連結 ストリング (ツールキットを使用しない場合)を戻します。

「テキストのアンラップ (Unwrap Text)」オプションを、ストリング・ディスクリ プターの「折り返し (Wrap)」オプションと混同しないでください。こちらのオプシ ョンは、フィールドでなくテキストの長方形ブロックを基にしています (70 ペー ジの『マクロ・ランタイムが長方形域を検索する方法 (折り返しオプション)』を参 照)。

「テキストのアンラップ (Unwrap Text)」が true で「連続抽出 (Continuous Extract)」が false の場合: 「連続抽出 (Continuous Extract)」が false の場合、 行と列の組はテキストの長方形ブロックの隅を表します。「テキストのアンラップ (Unwrap Text)」を true に設定すると、マクロ・ランタイムはテキストの長方形ブロックの各行を読み取り、行内の各フィールドを次のように処理します。

- フィールドが行の外側で始まり行の中に続く場合、マクロ・ランタイムはフィールドを無視する。
- フィールドが行の中で始まり行の中で終わる場合、マクロ・ランタイムはフィー ルドの内容を結果に含める。
- フィールドが行の中で始まり行の外側で終わる場合、マクロ・ランタイムはフィ ールド全体の内容 (テキストの長方形ブロックの外側にある部分を含む)を結果 に含める。

「テキストのアンラップ (Unwrap Text)」オプションの目的は、フィールドが行と 行の間で折り返している場合でも、フィールドの内容全体を 1 つのストリングとし てキャプチャーすることです。

例えば、セッション・ウィンドウの幅が 80 文字で、セッション・ウィンドウの行 9、10、11、および 12 が次のとおりであるとします。

.....Compress or print data set...... Reset statistics...... Catalog or display information of an entire data set.....

さらに、上記の行の中で次のテキスト域がフィールドであるとします。

Compress or print data set Reset statistics Catalog or display information of an entire data set

最後に、次のように仮定します。

- 「連続抽出 (Continuous Extract)」は false (これはデフォルト設定値です)。
- 「テキストのアンラップ (Unwrap Text)」は true。
- マーキング長方形の左上隅は行 9、列 63 ('print' の 'n')、右下隅は行 11、列 73 ('or' の後の ' ')。

- 抽出名は 'Extract1'。
- データ・プレーンは TEXT_PLANE。

IBM Host Access toolkit を使用している場合、マクロ・ランタイムは toolkit メソ ッドの戻り値として、ストリングの配列 'Reset statistics'、'Catalog or display information of an entire data set' を戻します。マクロ・ランタイム は、次の処理を行います。

- フィールドがマーキング長方形の外側で始まっているので、 'Compress or print data set' をスキップする。
- フィールドがマーキング長方形の中で始まっているので、 'Reset statistics' を戻す。
- フィールドがマーキング長方形の中で始まっているので、 'Catalog or display information of an entire data set' を戻す (このフィールドが次の行に折り返 していても)。

toolkit を使用していない場合、マクロ・ランタイムは個々のストリングを連結し、 単一のストリングとして「抽出 (Extract)」ウィンドウで指定した変数に格納しま す。この例では、マクロ・ランタイムはストリング 'Reset statisticsCatalog or display information of an entire data set' を変数に格納します。

「テキストのアンラップ (Unwrap Text)」が true で「連続抽出 (Continuous Extract)」が true の場合: 「連続抽出 (Continuous Extract)」が true の場合、2 組の行と列は、行間で折り返す (必要な場合) 連続したテキスト・シーケンスの開始 位置と終了位置を表します。このときに「テキストのアンラップ (Unwrap Text)」 を true に設定すると、マクロ・ランタイムは連続したテキスト・シーケンスを次 のように処理します。

- フィールドがシーケンスの外側で始まりシーケンスの中に続く場合、マクロ・ランタイムはフィールドを無視する。
- フィールドがシーケンスの中で始まりシーケンスの中で終わる場合、マクロ・ランタイムはフィールドの内容を結果に含める。
- フィールドがシーケンスの中で始まりシーケンスの外側で終わる場合、マクロ・ ランタイムはフィールド全体の内容 (テキストの長方形ブロックの外側にある部 分を含む)を結果に含める。

Toolkit を使用したデータ・プレーンからのデータのキャプチャー

IBM Host Access Toolkit 製品の Java API を使用して、TEXT_PLANE など任意 のデータ・プレーンからのデータにアクセスできます。

データ・プレーンと各プレーンに関連したデータのタイプは、次のとおりです。

- TEXT_PLANE 画面に表示される文字
- FIELD_PLANE 3270 または 5250 フィールド属性
- COLOR_PLANE 3270 または 5250 カラー属性
- EXFIELD_PLANE 3270 または 5250 拡張属性
- DBCS_PLANE 2 バイト文字セットの文字
- GRID_PLANE 2 バイト文字セットのグリッド情報

Toolkit を使用して抽出したデータにアクセスするには、 MacroRuntimeListener クラスをインプリメントし、マクロ Bean にユーザー自身を登録する必要がありま す。抽出アクションが実行されるたびに、マクロ Bean はユーザーの macroExtractEvent() メソッドを呼び出して、データをユーザーに発信します。デー タにアクセスするには、MacroExtractEvent の get メソッドを使用します。

セッションの指定

「ホスト ID (Host ID)」フィールドを使用して、このアクションを実行するセッションを指定します。

- マクロを起動するセッションでこのアクションを実行する場合は、「ホスト ID (Host ID)」フィールドをブランクのままにする。
- 別のセッションでこのアクションを実行する場合は、「ホスト ID (Host ID)」 フィールドにセッション参照を入力する(196 ページの『ホスト ID の指定』を 参照)。

参照されたセッションがアクティブである場合は、このセッションに任意の自動編 集機能を使用できます (197 ページの『異なるセッションでの自動編集機能の使 用』を参照)。

FileUpload アクション (<fileupload> エレメント)

FileUpload アクションは、ホスト・データベース内のテーブルの作成、テーブルへ のデータの追加、テーブルの置換、テーブルの更新を行える強力で有用なアクショ ンです。これは、ホスト・データベースに SQL ステートメントを送信する SQLQuery アクションと対になるアクションです (121 ページの『SQLQuery ア クション (<sqlquery> エレメント)』を参照)。

FileUpload アクションは、マクロをサポートする任意のタイプの Host On-Demand セッションで使用できます (3270 ディスプレイ、5250 ディスプレ イ、VT ディスプレイ、または CICS ゲートウェイ)。

接続先のデータベース・サーバーは、エミュレーター・セッションを実行している ホストとは別のホストにあってもかまいません。

データベース・サーバーとの接続

データベースの URL: 「データベース URL (Database URL)」フィールドに、デ ータベースへのアクセスを提供するデータベース・サーバーの URL を入力しま す。データベース URL の形式は、データベースへのアクセスに使用する Java Database Connectivity (JDBC) ドライバーのタイプによって異なります (JDBC ド ライバーについて詳しくは、 94 ページの『ドライバー ID とドライバー・クラ ス』を参照)。表 16 は、Host On-Demand に組み込まれている JDBC ドライバー のデータベース URL を示しています。

表 16. データベース URL の形式

Host On-Demand で	データベース URL の形式:	例えば、次のとおりです。
提供されるドライバー :		
AS/400 Toolbox for Java	jdbc:as400://[host]	• jdbc:as400://myISeries
		• jdbc:as400://9.88.24.163

実際のデータベース URL (jdbc:as400://myISeries など) では、上記の形式例に示 されている大括弧は使用しない でください。

リモート・サーバーは、Host On-Demand エミュレーター・セッションの接続先の ホストとは別のホストに配置することができます。例えば、FileUpload アクション は iSeries ホストを指定できます。これは、同じ FileUpload アクションが、 zSeries ホストに接続されている 3270 ディスプレイ・セッションで実行中のマクロ の一部であっても同様です。

Host On-Demand で提供されるドライバー以外の JDBC ドライバーを使用する場合、データベース URL の正しい形式については、ドライバー・ベンダーの提供する資料を調べてください。

ドライバー ID とドライバー・クラス: FileUpload アクションがデータベースに アクセスするために使用する JDBC ドライバーは、Java クライアント・パッケー ジです。これは、リモート・ホスト上のサーバー・プログラムと通信する Host On-Demand クライアント・ワークステーション上にあります。リモート・ホスト 上のこのサーバー・プログラムを使用すると、データベースへのアクセスを行うこ とができます。

Host On-Demand クライアントには、AS/400 Toolbox for Java の JDBC ドライ バーがすでに組み込まれています。このドライバーは、エミュレーター・クライア ントの一部として自動的にダウンロードされます。このドライバーにより、クライ アントは、正しく構成された iSeries または AS/400 上の、DB2/400 データにアク セスできます。

別の JDBC ドライバーが必要な場合は、リモート・データベース・サーバーの管理 者と連絡を取ってドライバーを取得してください。ドライバーを Host On-Demand クライアント・ワークステーションにデプロイするには、いくつかのアクションを 実行する必要があります。

マクロ・エディターの「FileUpload アクション (FileUpload action)」ウィンドウの 「ドライバー ID (Driver Identifier)」リスト・ボックスで Host On-Demand で提 供されるドライバーを選択するか、別のドライバーを使用する場合は「その他 (0ther)」を選択します。

「ドライバー・クラス (Driver Class)」フィールドには、ドライバーの完全修飾 Java クラス名が入ります。Host On-Demand で提供されるドライバーを選択する 場合、マクロ・エディターは「ドライバー・クラス (Driver Class)」フィールドに クラス名 (com.ibm.as400.access.AS400JDBCDRIVER) を表示します。このクラス名を 変更することはできません。一方、「ドライバー ID (Driver Identifier)」リスト・ ボックスで「その他 (Other)」を選択する場合は、「ドライバー・クラス (Driver Class)」フィールドにドライバーの完全修飾クラス名を入力する必要があります。完 全修飾クラス名が分からない場合は、ドライバーのプロバイダーにお問い合わせく ださい。名前を入力する場合は、大/小文字に注意してください (例えば、com と COM は異なります)。

「ドライバー ID (Driver Identifier)」リスト・ボックスの選択可能なドライバーの リストにドライバーを追加する場合 (毎回「その他 (Other)」を選択してクラス名
を入力しないようにする場合) は、ドライバーを Host On-Demand に登録できま す (オンライン・ヘルプの『JDBC ドライバーの登録 (Registering a JDBC driver)』を参照)。

ユーザー ID とパスワード: データベース接続でユーザー ID とパスワードが必要 な場合は、「ユーザー ID (User ID)」フィールドにユーザー ID、「パスワード (Password)」フィールドにパスワードを入力します。

Host On-Demand は、「パスワード (Password)」フィールドに入力するキー・シ ーケンスを暗号化します。この暗号化は、入力アクションで「パスワード (Password)」チェック・ボックスを選択する場合に使用される暗号化とまったく同 じ方法で機能します (99 ページの『パスワード』を参照)。要確認:

- 「パスワード (Password)」フィールドにパスワード (mypass など) を入力する 場合、マクロ・エディターはパスワードをアスタリスク (******) で表示する。
- 入力フォーカスを別の入力フィールドに移動すると、マクロ・エディターでは次のようになる。
 - 1. 暗号化されたパスワードが生成される (q0eqOskTUBQ= など)。
 - 2. 「パスワード (Password)」フィールドに、アスタリスクを使用した暗号化さ れたパスワード (**********) が表示される。(暗号化されたパスワードの 実際の文字は、コード・エディターで確認できます。)
- パスワードはストリングである。したがって、拡張マクロ形式を使用する場合は、パスワードを単一引用符で囲んで入力してください(例えば、'mypass')。マクロ・エディターは、単一引用符を含めたストリング全体を暗号化します。
- マクロ・エディターによってパスワードが暗号化された後でパスワードを変更する必要がある場合は、新規パスワードを入力する前に、フィールドのすべての文字を完全に削除する必要がある。

ファイル・アップロードの情報の指定

ファイルのアップロードに必要な接続情報 (前述のデータベース URL、ドライバー ID、ドライバー・クラス、ユーザー ID、およびパスワード) に加えて、特定のファ イル・アップロード操作についての情報も提供する必要があります。この情報に は、次のものが含まれます。

- 変更するホスト・データベース内のテーブルの名前。
- ホスト・データベース内のテーブルに追加するデータが含まれているローカル・ ファイルの名前。
- 実行するファイル・アップロードのタイプ (作成、置換、付加、または更新)。

また、2 つのタイプのファイル・アップロードには追加情報 (作成操作の場合はフィールド記述テーブル、更新操作の場合はキー欄)を指定する必要があります。表 17 に、この情報の要約を示します。

 ファイル・アッ プロードのタイ ルの名前:
 ホスト・テーブ ルの名前:
 ソース・ファイ ルの名前:
 その他の情報:

 プ:
 必要
 必要
 フィールド記述テーブル

 置換
 必要
 必要
 (なし)

表 17. ファイル・アップロードのタイプと必要な情報

表 17. ファイル・アップロードのタイプと必要な情報 (続き)

ファイル・アッ プロードのタイ プ:	ホスト・テーブ ルの名前:	ソース・ファイ ルの名前 :	その他の情報:
付加	必要	必要	(なし)
更新	必要	必要	キー欄

ファイル名およびファイル・タイプ: 「ファイル名」フィールドに、ホスト・デー タベース内のテーブルに追加するデータが含まれているローカル・ファイルのパス を入力します。 (ファイル・パスはストリングであるため、適切なマクロ形式 (基本 または拡張) でストリングを入力してください)。または、「ブラウズ (Browse)」を クリックしてファイルを参照します。注: このフィールドには、ドライブ名 (ある場 合) を含めた完全なファイル・パスおよびファイル名を指定する必要があります (例 えば、c:¥Documents and Settings¥User1¥input.txt)。

「ファイル・タイプ」リスト・ボックスで、ファイルのタイプをクリックします。 このリスト・ボックスには、Database On-Demand クライアントでサポートされる のと同じデータベース・ファイル・タイプがリストされます。

テーブル名: 「テーブル名 (Table Name)」フィールドに、更新するホスト・デー タベース内のテーブルの名前 (hodtest など) を入力します。ホスト・データベース にスキーマがある場合は、テーブル名に加えてスキーマ名も含めてください (hod.hodtest)。

作成: 新規テーブルを作成するには、「アップロードのタイプ (Upload Type)」リ スト・ボックスを展開して「作成 (create)」をクリックします。

「フィールド記述テーブル (Field Description Table)」フィールドに、データベー ス・サーバーが新規テーブルの列 (フィールド) 名および列幅を読み取るホスト・デ ータベースのスキーマおよびテーブルの名前 (hod.hodmodel など) を入力します。 このテーブルには、作成するテーブルと同じ数の列が含まれている必要がありま す。また、このテーブルの列名および列幅は、新規テーブルで使用するものと同じ でなければなりません。

同様に、「ファイル名」フィールドに指定するファイルの列数および列幅は、作成 するテーブルと同じでなければなりません。

マクロ・ランタイムが作成操作を実行すると、データベース・サーバーは、指定し たファイルのデータと指定したフィールド記述テーブルの列名および列幅を使用し て、新規テーブルを作成します。

置換: 既存のテーブルの内容を置換するには、「アップロードのタイプ (Upload Type)」リスト・ボックスを展開して「置換 (replace)」をクリックします。

「ファイル名」フィールドに指定するファイルの列数および列幅は、内容を置換す るテーブルと同じである必要があります。

マクロ・ランタイムが置換操作を実行すると、データベース・サーバーは、指定したファイルのデータと既存のテーブルにすでに定義されている列名および列幅を使用して、指定したテーブルの内容を置換します。例えば、テーブル hodtest には、

置換操作前に 8 列と 300 行のデータが含まれていますが、置換操作後は 8 列と 250 行 (すべて新規データ)のデータだけを含むことができます。

付加: 既存のテーブルの最後に行を付加するには、「アップロードのタイプ (Upload Type)」リスト・ボックスを展開して「付加 (append)」をクリックしま す。

「ファイル名」フィールドに指定するファイルの列数および列幅は、付加の対象と なるテーブルと同じである必要があります。

マクロ・ランタイムが付加操作を実行すると、データベース・サーバーはデータの 行を指定したファイルから指定したテーブルの最後に付加します。例えば、テーブ ル hodtest には、付加操作前に 8 列と 300 行のデータが含まれていますが、付加 操作後は 8 列と 320 行のデータ (元の 300 行に 20 行が付加) が含まれます。

更新: 既存のテーブルの一部を選択的に更新するには、「アップロードのタイプ (Upload Type)」リスト・ボックスを展開して「更新 (update)」をクリックしま す。

「キー欄 (Key Columns)」フィールドに、更新する列の名前を入力します。

「ファイル名」フィールドに指定するファイルのデータ列数は、「キー欄 (Key Columns)」フィールドに指定する名前の数と同じである必要があります。例えば、「キー欄 (Key Columns)」フィールドに 3 つの列名を入力する場合は、テーブル には 3 つのデータ列が含まれている必要があります。

マクロ・ランタイムが更新操作を実行すると、データベース・サーバーは指定した データ列を指定したファイルのデータで置換します。例えば、テーブル hodtest に 8 行と 250 列が含まれている場合は、テーブルの最初の列、2 番目の列、および 7 番目の列を更新できます。

Host On-Demand クライアント用のデータベース・コンポーネント のプリロード

デプロイメント・ウィザードで、データベースにアクセスするために Host On-Demand クライアントがプリロードするマクロ・コンポーネントを指定できま す。オンライン・ヘルプの『プリロード・オプション (Features1) (Preload Options (Features1))』を参照してください。

入力アクション (<input> エレメント)

入力アクションは、キー・ストロークのシーケンスをセッション・ウィンドウに送 ります。シーケンスには、文字を表示するキー (a、b、c、#、& など) だけでな く、アクション・キー ([enterreset]、[copy]、[paste] など) も含めることができま す。

このアクションは、実際のユーザーからのキーボード入力をシミュレートします。

タイプ入力の開始位置

「行 (Row)」フィールドと「列 (Column)」フィールドを使用して、入力シーケン スを開始するセッション・ウィンドウ内の行と列の位置を指定します。例えば、入 力アクションに行 23、列 17 を指定し、入力アクションのストリング値として Hello world を指定した場合、(指定した位置が入力フィールド内にあると仮定して) マクロ・ランタイムは、キー・シーケンス Hello world をセッション・ウィンドウ に行 23、列 17 から入力します。

行または列の位置を 0 に指定した場合、入力アクションが実行されると、マクロ・ ランタイムはセッション・ウィンドウのテキスト・カーソルがある実際の行と列の 位置からキー・シーケンスを入力します。テキスト・カーソルの位置が関係ないコ ンテキストの場合 (例えば、[copy] アクション・キー)、またはテキスト・カーソル の位置が予測可能な場合 (例えば、直前のマウス・クリック・アクションによって テキスト・カーソルが特定の位置に移動した場合や、アプリケーションがアプリケ ーション画面を表示するときにテキスト・カーソルの位置を指定した場合)を除い ては、行または列に 0 を指定しないでください。

入力エラー

マクロの再生中に、セッション・ウィンドウは実際のユーザーがキーを入力した場 合と同様にキー入力エラーに対応します。

例えば、入力アクションが文字を表示するキー (a、b、c、#、& など) を送った場 合に、テキスト・カーソルが 3270 または 5250 入力フィールドの中になければ、 セッションはキー入力を禁止し、オペレーター情報域にエラー・シンボルを表示し ます。この対応は、実際のユーザーが入力したキー・ストロークに対するものとま ったく同じです。

入力ストリング

「ストリング (String)」フィールドは、アクションに実行させるキー・シーケンス を指定するための入力フィールドです。

文字を表示するキー (a、b、c、#、& など) を指定するには、そのキー自体を入力 します。

キーを「アクション・キー (Actions Keys)」リスト・ボックスから指定するには、 目的のキー ([backspace] など) までリストをスクロールし、「アクション・キーの 挿入 (Insert Action Key)」をクリックします。「ストリング (String)」フィールド の次の入力位置に、キーの名前が大括弧で囲まれて表示されます。「アクション・ キー (Action Keys)」リスト・ボックスにあるキーは、アルファベット順にリスト されていないので注意してください。必要なキーを見つけるために、リストをスク ロールし続ける必要が生じることがあります。

別の方法として、単に名前自体を大括弧で囲んで「ストリング (String)」入力フィ ールドに入力する (例: [backspace]) ことにより、アクション・キーを指定するこ ともできます。

3270 ディスプレイ・セッションの「アクション・キー (Action Keys)」リストに は、次のコピー・アンド・ペースト・キーが表示されます。

[copy]	[mark right]
[copyappend]	[mark up]
[cut]	[paste]
[mark down]	[pastenext]
[mark left]	[unmark]

その他のキーについては、 267 ページの『入力アクションの略号キーワード』を参 照してください。

ホスト・アクション・キーの変換

「ホスト・アクション・キーの変換 (Translate Host Action Keys)」フィールド は、入力シーケンスにあるアクション・キー名 ([copy]、[enterreset]、[tab] など) をアクション・キーとして解釈するか、リテラル文字シーケンスとして解釈するか をマクロ・ランタイムに指示します。デフォルトは true です (アクション・キー名 をアクション・キーとして解釈)。

例えば、入力キー・シーケンスが '[up][up]Hello world' で、テキスト・カーソル が行 4、列 10 にあるとします。「ホスト・アクション・キーの変換 (Translate Host Actions Keys)」の値が true の場合は、この入力シーケンスを実行すると、 マクロ・ランタイムはテキスト・カーソルを上方向に 2 行移動した後、行 2、列 10 から Hello world を入力します。一方、「ホスト・アクション・キーの変換 (Translate Host Actions Keys)」の値が false の場合、マクロ・ランタイムは行 4、列 10 から [up][up]Hello World を入力します。

カーソルを入力の最後に移動

「ホスト・アクション・キーの変換 (Translate Host Action Keys)」フィールドが true (デフォルト) に設定されている場合、マクロ・エディターは「カーソルを入力 の最後に移動 (Move Cursor to End of Input)」リスト・ボックスも true に設定 し、このリスト・ボックスを使用不可にします。リスト・ボックスが使用不可にな っていても、値は true に設定されています。

「ホスト・アクション・キーの変換 (Translate Host Action Keys)」リスト・ボッ クスを false に設定した場合は、「カーソルを入力の最後に移動 (Move Cursor to End of Input)」リスト・ボックスが使用可能になり、このリスト・ボックスを false、 true、または実行時に評価される式に設定できます。

リスト・ボックスの値が true (デフォルト) の場合、マクロ・ランタイムは、実際 のユーザーがキーボード入力を行っている場合と同じようにテキスト・カーソルを 移動します。例えば、キーが 'a' などのテキスト文字の場合、マクロ・ランタイム は文字をセッション・ウィンドウに入力し、テキスト・カーソルを 'a' の後にある 最初の文字位置に移動します。同様に、キーが [tab] の場合、マクロ・ランタイム はテキスト・カーソルを次のタブ位置に移動します。

これに対し、「カーソルを入力の最後に移動 (Move Cursor to End of Input)」リ スト・ボックスが false の場合、マクロ・ランタイムはテキスト・カーソルをまっ たく移動しません。テキスト・カーソルは、マクロ・ランタイムが入力アクション を実行する前にあった同じ位置に残ります。

パスワード

「パスワード (Password)」チェック・ボックスを使用して、「ストリング (String)」フィールドの入力キー・シーケンスを暗号化できます。「ストリング (String)」フィールドの内容を暗号化すると、Host On-Demand は暗号化された入 カキー・シーケンスのバージョンをマクロ・スクリプト (<input> エレメント) に格 納し、プレーン・テキスト (暗号化されていない) 入力シーケンスのバージョンは保 存しません。 例えば、図 32 は、入力キー・シーケンスが暗号化されていない ('myPassword') <input> エレメントを示しています。

図 32. 入力キー・シーケンスが暗号化されていない <input> エレメント

一方、図 33 は、入力キー・シーケンスが暗号化された (I7xae6rSV1VFF6qzhWRfKw==) 同じ <input> エレメントを示しています。この例で は、<input> エレメントの encrypted 属性が true に設定されていることに注意し てください。

input value="I7xae6rSV1VFF6qzhWRfKw==" row="20" col="16" movecursor="true" xlatehostkeys="true" encrypted="true" />

図 33. 入力キー・シーケンスが暗号化された <input> エレメント

マクロ・エディターでは、暗号化された入力キー・シーケンスはアスタリスクで表示されます (例えば、暗号化された 'myPassword' は「ストリング (String)」フィー ルドに I7xae6rSV1VFF6qzhWRfKw== ではなく、************************ と表示されます)。

暗号化により、機密データを不用意に公開することなく、入力アクションに機密デ ータ (パスワードなど)を含めることができます。許可されていないユーザーは、テ キスト・エディター、マクロ・エディター、またはコード・エディターでマクロ・ スクリプトを表示しても機密データを見ることはできません。

入力キー・シーケンスの暗号化後、Host On-Demand は、マクロ・エディターまた はコード・エディターを使用して入力キー・シーケンスを暗号化解除することを誰 にも許可しません。 Host On-Demand は、マクロ・ランタイムがマクロ再生中に 入力アクションを処理するまで、入力アクションの入力キー・シーケンスを暗号化 解除しません。マクロ・ランタイムが入力アクションを処理すると、マクロ・ラン タイムは暗号化された入力キー・シーケンスを暗号化解除し、暗号化されていない 入力キー・シーケンスをセッション・ウィンドウの指定された行および列の位置に 入力します。

一般に 3270 または 5250 環境では、パスワードなどの機密入力キー・シーケンス
 用に、ホスト・アプリケーションは入力キー・シーケンスの宛先として非表示の入
 カフィールドを作成し、プレーン・テキストではなくブランクまたはアスタリスク
 (*) が表示されるようにします。

ただし、マクロ・スクリプトを不用意に公開すると、機密漏れの危険性がありま す。マクロ・スクリプトのコピーにアクセスできれば、巧妙な手口で元の暗号化さ れていない入力キー・シーケンスが見られる可能性があります。つまり、入力アク ションの「行 (Row)」「列 (Column)」フィールドを編集して、マクロ再生中にマ クロ・ランタイムが暗号化解除された入力キー・シーケンスを通常の表示可能なフ ィールドに入力するようにすることができます。 より高いセキュリティーを実現するには、入力アクションではなくプロンプト・ア クションを使用します。プロンプト・アクションでは、入力キー・シーケンスは暗 号化された形式によっても、マクロ・スクリプトに格納されることはありません。 マクロ再生中にマクロ・ランタイムがプロンプト・アクションを処理する場合、マ クロ・ランタイムは、入力フィールドと、入力キー・シーケンスを入力するようエ ンド・ユーザーに促すメッセージが表示されたウィンドウをポップアップします。 エンド・ユーザーが入力キー・シーケンスを入力して「OK」をクリックすると、マ クロ・ランタイムはポップアップ・ウィンドウを除去し、入力キー・シーケンスを セッション・ウィンドウの指定した行および列の位置に送信します。

入力アクションを使用する場合は、エンド・ユーザーが介入しなくてもマクロ再生 中にマクロ・スクリプトが自動的に実行されるという利点があります。ただし、機 密データが変更される場合 (例えば、パスワードが失効して新規の別のパスワード が必要になる場合) は、入力アクションを新規入力キー・シーケンスで更新する必 要があります。

マクロ記録中の自動暗号化: マクロ記録中に、「パスワードの記録」オプション は、3270 または 5250 の非表示のフィールドに入力される入力キー・シーケンスを Host On-Demand が記録する方法を制御します (3270 ディスプレイ・セッション および 5250 ディスプレイ・セッションの場合のみ)。

「パスワードの記録」が使用可能になっている場合 (デフォルト)、Host On-Demand は、暗号化された入力キー・シーケンスを入力アクションとして自動 的に記録します。「パスワードの記録」が使用不可になっている場合、Host On-Demand は、入力キー・シーケンスをプロンプト・アクションとして記録しま す。詳しくは、 189 ページの『パスワードの記録』を参照してください。

「パスワード」チェック・ボックスの使用: マクロ記録中に入力キー・シーケンス が自動的に暗号化される場合、マクロ・エディターで入力アクションを表示する と、「パスワード (Password)」チェック・ボックスが選択されていて (チェックマ ークが付いている)、暗号化されたキー・シーケンスを表すいくつかのアスタリスク (******) が「ストリング (String)」フィールドに表示されています。

一方、マクロ記録中に入力キー・シーケンスが自動的に暗号化されない場合 (原因 としてセッションが 3270 または 5250 セッションではない、または入力フィール ドが非表示の入力フィールドであることが考えられる)、「パスワード」チェック・ ボックスはクリアされていて (チェックマークが付いていない)、暗号化されていな い入力キー・シーケンスが「ストリング (String)」フィールドに表示されていま す。

マクロ記録中に入力キー・シーケンスが自動的に暗号化されない場合は、マクロ・ エディターで暗号化することができます。入力キー・シーケンスを暗号化するに は、次のステップを実行します。始める前に、「パスワード」チェック・ボックス がクリアされていない場合はクリアしてください (チェックマークを外す)。

- 希望の入力キー・シーケンスが「ストリング (String)」フィールドに表示されて いない場合は、入力キー・シーケンスを「ストリング (String)」フィールドに入 力する。
 - 入力キー・シーケンスが「ストリング (String)」フィールドに正常に表示される (例えば、'myPassWord')。

- 拡張マクロ形式を使用する場合は、入力キー・シーケンスを単一引用符で囲む ('myPassWord')。
- 2. 「パスワード」チェック・ボックスを選択する。
 - マクロ・エディターが入力キー・シーケンスを暗号化し、入力キー・シーケンスがアスタリスク (********************************) を使用して「ストリング (String)」フィールドに表示される。

暗号化された入力キー・シーケンスを作成して、「ストリング (String)」フィール ドへの入力時に入力キー・シーケンスを暗号化されていない形式で表示しない場合 は、次の方法を使用します。

- 1. 「ストリング (String)」フィールドが空になっていない場合は、フィールドをク リアする。
- 2. 「パスワード」チェック・ボックスを選択する。
- 3. 「ストリング (String)」フィールドに入力キー・シーケンスを入力する。
 - 拡張マクロ形式を使用する場合は、入力キー・シーケンスを単一引用符で囲む ('myPassWord')。
 - 「ストリング (String)」フィールドに入力すると、マクロ・エディターはア スタリスクを使用して文字を表示する ('myPassword' は *********** と表 示される)。
 - 入力フォーカスを「ストリング」フィールドから移動すると(つまり、別の フィールドをクリックすると)、マクロ・エディターは入力キー・シーケンス を暗号化する。

入力キー・シーケンスの暗号化後に、入力キー・シーケンスを暗号化しないように するか、訂正することができます。

入力キー・シーケンスの暗号化後に、入力キー・シーケンスを暗号化しないように するには、次のステップを実行します。

- 1. 「パスワード (Password)」チェック・ボックスをクリアする。
 - マクロ・エディターは暗号化されたストリングを破棄し、「ストリング (String)」フィールドをクリアする。
 - 何らかの理由で「ストリング (String)」フィールドがクリアされない場合 は、Backspace キーまたは Delete キーを使用してフィールドの文字を削除 する。
- 2. 「ストリング (String)」フィールドに暗号化されていないキー・シーケンスを入 力する。

入力キー・シーケンスの暗号化後に、入力キー・シーケンスを訂正するには、次の ステップを実行します。

- 1. Backspace キーまたは Delete キーを使用して「ストリング (String)」フィール ドをクリアする。
 - 暗号化された入力キー・シーケンス全体を削除してフィールドを空にする。
- 「ストリング (String)」フィールドに訂正した入力キー・シーケンスを入力する。
 - 拡張マクロ形式を使用する場合は、入力キー・シーケンスを単一引用符で囲む ('myPassWord')。

- 「ストリング (String)」フィールドに入力すると、マクロ・エディターはア スタリスクを使用して文字を表示する ('myPassword' は *********** と表 示される)。
- 入力フォーカスを「ストリング」フィールドから移動すると(つまり、別の フィールドをクリックすると)、マクロ・エディターは入力キー・シーケンス を暗号化する。

「ストリング (String)」フィールド内のアスタリスクのストリング (*******) を上 書きしたり、ストリングに文字を挿入することによって、暗号化された入力キー・ シーケンスを訂正しないでください。そのようにすると、暗号化されていない訂正 によって暗号化された入力キー・シーケンスが破壊されます。その後、マクロ・エ ディターは、暗号化されていないストリングが入力されたものと見なし、破壊され たシーケンスを再暗号化します。その結果、マクロ再生中にマクロ・ランタイムが 入力アクションを処理すると、暗号化解除されたシーケンスは期待したものとは異 なる入力キー・シーケンスになります。(また、拡張マクロ形式を使用しており、破 壊された入力キー・シーケンスを単一引用符で囲まない場合には、マクロ・エディ ターはエラー・メッセージを生成します。)

コード・エディターの使用: コード・エディターは、マクロ・エディターと同じ規 則に従って入力キー・シーケンスを暗号化します。

コード・エディターでは、常に次のいずれかのアクションを実行できます。

- 入力キー・シーケンスを暗号化する新規の <input> エレメントを、編集可能テ キスト域に入力する。
- 入力キー・シーケンスを暗号化する <input> エレメントを、システムのクリッ プボードから編集可能テキスト域に貼り付ける。

また、コード・エディターの使用中に、encrypted 属性 (暗号化を活動化または非 活動化する) の値を true から false、または false から true に変更することもで きます。

ただし、コード・エディターを使用して value 属性 (暗号化されたまたは暗号化さ れていない入力キー・シーケンスを含む) の値を変更する場合で、encrypted 属性 が true に設定されている場合は、暗号化された入力キー・シーケンスを完全に削 除してから (value="" にする)、暗号化する新規入力キー・シーケンスを入力しま す。

value 属性の暗号化された入力キー・シーケンスを上書きしたり、キー・シーケン スに文字を挿入することによって、暗号化された入力キー・シーケンスを訂正しな いでください。そのようにすると、暗号化されていない訂正によって暗号化された 入力キー・シーケンスが破壊されます。

変数名の暗号化: 変数名 (\$var1\$ など) を、マクロ・エディターで「ストリング (String)」フィールドに (または、コード・エディターで value 属性の値の部分に) 入力して、変数の名前を暗号化できますが (通常の入力キー・シーケンスを暗号化 する場合と同じステップを使用する)、通常これは実用的ではありません。変数名を 暗号化すると、変数名を構成する文字だけが暗号化されるからです。変数自体の内 容は暗号化されません。 マクロ再生中に、マクロ・ランタイムは、暗号化されたテキストを暗号化解除して プレーン・テキスト (\$var1\$) を取得し、プレーン・テキストが変数名であることを 認識してから、通常の方法で変数を評価します。

セッションの指定

「ホスト ID (Host ID)」フィールドを使用して、このアクションを実行するセッションを指定します。

- マクロを起動するセッションでこのアクションを実行する場合は、「ホスト ID (Host ID)」フィールドをブランクのままにする。
- 別のセッションでこのアクションを実行する場合は、「ホスト ID (Host ID)」 フィールドにセッション参照を入力する(196 ページの『ホスト ID の指定』を 参照)。

参照されたセッションがアクティブである場合は、このセッションに任意の自動編 集機能を使用できます (197 ページの『異なるセッションでの自動編集機能の使 用』を参照)。

例

105 ページの『コピーと貼り付けの例』を参照してください。

メッセージ・アクション (<message> エレメント)

メッセージ・アクションは、タイトル、メッセージ、および「OK」ボタンがあるポ ップアップ・ウィンドウを表示します。マクロ・ランタイムは、ユーザーが「OK」 をクリックするまでアクションを終了しません。

このメッセージは、次のようにさまざまなシナリオで使用できます。

- ユーザーに対して指示、エラー・メッセージ、または状況メッセージを表示する。
- ユーザーが何らかのアクションを実行するまでマクロの実行を中断する。
- デバッグ用の値を表示する。

メッセージ表題とメッセージ・テキストの表示

メッセージ・ウィンドウの表題バーに表示する表題は、「メッセージ表題 (Message Title)」入力フィールドに指定します。

メッセージ・ウィンドウの中に表示するテキストは、「メッセージ・テキスト (Message Text)」入力フィールドに指定します。

両方の入力フィールドは入力値としてストリングを予期するので、ストリングとし て評価される任意のエンティティーを指定できます(41 ページの『等価』を参 照)。演算式を使用する場合は、式自体に即時値、変数、演算式、および Java メソ ッドの呼び出しを含めることができます(37 ページの『算術演算子および式』を 参照)。

また、データ型変換規則(40ページの『自動データ型変換』を参照)とストリング 連結記号(38ページの『ストリング連結演算子(+)』を参照)を使用することもで きます。例えば、\$intResult\$ という名前の整変数の値を表示する場合は、「メッ セージ・テキスト (Message Text)」入力フィールドに次のストリングを指定できま す。

'The result is ' + \$intResult\$ + '.'

\$intResult\$ の値が 204 ならば、マクロ・ランタイムはメッセージ・ボックスに次のテキストを表示します。 The result is 204.

マウス・クリック・アクション (<mouseclick> エレメント)

マウス・クリック・アクションは、ユーザーによるセッション・ウィンドウのマウ ス・クリックをシミュレートします。実際のマウス・クリックと同様に、クリック が行われたときにマウス・アイコンが指していた行と列の位置にテキスト・カーソ ルがジャンプします。

行と列の指定

「アクション (Actions)」ウィンドウの下部領域で、マウス・クリックを発生させる セッション・ウィンドウ上の行と列の位置を指定します。または、セッション・ウ ィンドウ自体をクリックすると、テキスト・カーソルの新しい位置を反映して、マ クロ・エディターの「行 (Row)」フィールドと「列 (Column)」フィールドの値が 更新されます。

セッションの指定

「ホスト ID (Host ID)」フィールドを使用して、このアクションを実行するセッションを指定します。

- マクロを起動するセッションでこのアクションを実行する場合は、「ホスト ID (Host ID)」フィールドをブランクのままにする。
- 別のセッションでこのアクションを実行する場合は、「ホスト ID (Host ID)」 フィールドにセッション参照を入力する(196 ページの『ホスト ID の指定』を 参照)。

参照されたセッションがアクティブである場合は、このセッションに任意の自動編 集機能を使用できます (197 ページの『異なるセッションでの自動編集機能の使 用』を参照)。

コピーと貼り付けの例

次の例では、セッション・ウィンドウ内でテキストのブロックにマークを付け、シ ステム・クリップボードにコピーして、セッション・ウィンドウ内の新しい位置に 再度貼り付ける方法を示します。この例では、ボックス選択アクション、入力アク ション、マウス・クリック・アクション、および休止アクションの各アクション・ エレメントを使用します。

このマクロ・スクリプトのテキストを本書からシステム・クリップボードにコピー し、システム・クリップボードからコード・エディターにコピーできます (204 ペ ージの『本書からコード・エディターへのスクリプトのコピー・アンド・ペース ト』を参照)。このスクリプトをマクロ・エディターに保管した後、マクロ・エディ ターまたはコード・エディターを使用してスクリプトを編集できます。 この例については、次の点に注意してください。

- この例は、COPY PASTE という名前の1 つのマクロ・スクリプト全体で構成されます。
- <actions> エレメント内では次のアクションが行われます。
 - <boxselection> アクションがマーキング長方形を描画する。
 - <pause> アクションが 1.5 秒待機し、マクロの再生時に何が行われているか ユーザーが確認できるようにする。
 - <input> アクションが、マークされた領域をクリップボードにコピーする [copy] アクションを入力する。
 - <mouseclick> アクションが、貼り付けを行う位置にカーソルを設定する。
 - <input> アクションが [paste] キーを入力する。このキーにより、クリップ ボードの内容がセッション・ウィンドウの新しい位置に貼り付けられます。
- このマクロは、ISPF 基本オプション・メニューから実行されるように書かれています(14ページの図 5 を参照)。マクロは、テキスト Spool Display and Search Facility を行 18 からシステム・クリップボードにコピーし、テキストをクリップボードから行 4 の「Option ===>」入力フィールドに貼り付けます。
- この例を実行しても正しく貼り付けが行われない場合は、指定したターゲット領域が 3270 または 5250 入力フィールドの中にあることを確認してください。
 Host On-Demand クライアントのアプリケーション画面内で、保護フィールドにテキストを貼り付けることはできません。

```
<HAScript name="COPY PASTE" description=" " timeout="60000" pausetime="300"</pre>
            promptall="true" author="" creationdate="" supressclearevents="false"
            usevars="false" >
   <screen name="Screen1" entryscreen="true" exitscreen="true"</pre>
            transient="false">
      <description>
         <oia status="NOTINHIBITED" optional="false" invertmatch="false" />
      </description>
      <actions>
         <boxselection type="SELECT" srow="18" scol="19"</pre>
                 erow="18" ecol="51" />
         <pause value="500" />
         <input value="[copy]" row="0" col="0" movecursor="true"
                 xlatehostkeys="true" encrypted="false" />
         <mouseclick row="4" col="15" />
         <input value="[paste]" row="0" col="0" movecursor="true"
                 xlatehostkeys="true" encrypted="false" />
      </actions>
      <nextscreens timeout="0" >
      </nextscreens>
  </screen>
```

</HAScript>

図 34. サンプル・コード COPY PASTE

休止アクション (<pause> エレメント)

休止アクションは、指定されたミリ秒数の間だけ待機した後、終了します。

具体的には、マクロ・ランタイムは <pause> エレメントを検出し、期間の値を読 み取って、指定されたミリ秒数だけ待機します。その後、マクロ・ランタイムは次 の項目の実行に進みます。

このアクションは次の場合に使用します。

- 何らかの状況で待ちを挿入したいとき。
- ホストがセッション・ウィンドウを更新するまで待つ。詳しくは、 147 ページの『画面の完了』を参照してください。
- デバッグの目的で遅延を追加する。

「期間 (Duration)」入力フィールドには、ミリ秒数を入力する必要があります。デフォルトは 10000 ミリ秒 (10 秒) です。

実行アクション (<perform> エレメント)

実行アクションは、インポートした Java クラスに属するメソッドを呼び出します (158 ページの『Java クラスのインポート型の作成』を参照)。

実行アクション以外にも、さまざまなコンテキストでメソッドを呼び出すことがで きます。ただし実行アクションは、例えば値を戻さないメソッドを呼び出したい場 合などのシナリオでは便利です。

実行アクションのほかに、メソッドを呼び出すことができるコンテキストは次のと おりです。

- 変数更新アクションを使用して、メソッドを呼び出し、変数に戻り値を割り当てることができます。戻り値を受け取る変数は、標準型に属する変数 (boolean、integer、string、double)、またはインポート型に属する変数 (例えば、Java クラス Object に基づく、インポート型 Object に属する変数 \$objTmp\$)のどちらかです。
- パラメーターのフィールドにメソッド呼び出しを指定すると、メソッドを呼び出して、マクロ・アクションの中で戻り値をパラメーターとして使用できます。例えば、抽出アクションの「行(Row)」パラメーターに、整数値を戻すメソッド呼び出しを使用できます。マクロ・ランタイムは、パラメーターがメソッド呼び出しであることを認識し、メソッドを呼び出して、整数の戻り値を「行(Row)」パラメーターの値として使用します。
- ・ 式の項としてメソッド呼び出しを使用すると、式の一部としてメソッドを呼び出すことができます。マクロ・ランタイムが式を評価し、その項がメソッド呼び出しであることを認識すると、メソッドを呼び出し、メソッドの値(例:ストリング)を項の値として使用します。
- メソッドを呼び出して、宣言した変数の初期値として戻り値を使用できます。

ー般に、実行アクションの外部では、メソッドから戻される値が有効であるすべて のコンテキストでメソッドを呼び出すことができます。

メソッドの呼び出し

「実行するアクション (Action to Perform)」フィールドにメソッド呼び出しを入力 します。変数と同じように、メソッド呼び出しをドル記号 (\$) で囲む必要がありま す (172 ページの『メソッド呼び出しの構文』を参照)。マクロ・ランタイムがメソ ッドを呼び出します。 172 ページの『マクロ・ランタイムが呼び出し先メソッド を検索する方法』も参照してください。

例

次の例では、実行アクションを使用してメソッドを呼び出す方法を示します。これ らの例については、次のことについて注意してください。

- 例 1 では、実行アクションは変数 \$importedVar\$ に対して update() アクショ ンを呼び出します。次の点に注意してください。
 - メソッド呼び出し全体はドル記号 (\$) で囲まれる。
 - メソッド呼び出しのコンテキストでは、変数名自体 (importedVar) はドル記号(\$) で囲まれない。
 - メソッドにパラメーターとして渡される変数は、通常どおりドル記号 (\$) で 囲む必要がある (\$str\$)。
 - メソッドにパラメーターとして渡されるストリングは、通常どおり単一引用
 符で囲む必要がある ('Application')。
- 例 2 では、実行アクションは静的メソッドを呼び出します。
- 例 3 では、実行アクションは変数が属するクラス (java.io.FileInputStream など) に属する close() メソッドを呼び出します。
- 例 4 では、実行アクションは変数が属するクラス (java.util.zip.ZipInputStream など) に属する createZipEntry() メソッドを呼び出します。
- 例 5 では、実行アクションは変数が属するクラス (java.util.Hashtable など) に 属する clear() メソッドを呼び出します。

```
<actions>
  <!-- Example 1 -->
  <perform value="$importedVar.update( 5, 'Application', $str$)$" />
  <!-- Example 2 -->
  <perform value="$MyClass.myInit('all')$" />
  <!-- Example 3 -->
  <perform value="$fip.close()$" />
  <!-- Example 4 -->
  <perform value="$zis.createZipEntry( $name$ )$" />
  <!-- Example 5 -->
  <perform value="$ht.clear()$" />
  </actions>
```

```
図 35. 実行アクションの例
```

PlayMacro アクション (<playmacro> エレメント)

PlayMacro アクションは、別のマクロを起動します。

マクロ・ランタイムが PlayMacro アクションを実行すると、現行マクロ (PlayMacro アクションを指定したマクロ) は終了し、ターゲット・マクロの指定さ れたマクロ画面の処理が開始されます。この処理はチェーニングと呼ばれ、呼び出 し側マクロをターゲット・マクロに「チェーニングする」といいます。呼び出し側 マクロへの戻りはありません。

PlayMacro アクションには、ターゲット・マクロの名前を指定する必要があり、またオプションで、マクロ・ランタイムが最初に処理するターゲット・マクロ内のマクロ画面の名前を指定します。

呼び出し側マクロからターゲット・マクロにすべての変数とその内容を転送するように、マクロ・ランタイムに指示できます。

PlayMacro アクションの追加

条件エレメントの外に追加する場合:

 PlayMacro アクションはマクロ・スクリプトに 1 つだけ追加でき、その PlayMacro アクションは、マクロ・スクリプトの「アクション (Actions)」リスト (<actions> エレメント)の最後のアクションでなければなりません。

条件エレメントの中に追加する場合:

- 「条件が真 (Condition is True)」ブランチ (<if>エレメント) には PlayMacro アクションを 1 つ追加でき、その PlayMacro アクションはブランチ (<if>エレ メント)の最後のアクションでなければなりません。
- また、「条件が偽 (Condition is False)」ブランチ (<else> エレメント) にも PlayMacro アクションを 1 つ追加でき、その PlayMacro アクションはブラン チ (<else> エレメント)の最後のアクションでなければなりません。

マクロには必要な数の条件エレメントを使用でき、それぞれの条件エレメントの 「条件は真 (Condition is True)」ブランチに 1 つの PlayMacro アクション、「条 件は偽 (Condition is False)」ブランチに 1 つの PlayMacro アクションを指定で きます。

マクロ・エディターとコード・エディターは、これらの規則を適用します。

ターゲット・マクロのファイル名と開始画面

「マクロ名 (Macro Name)」フィールドを使用して、ターゲット・マクロの名前を 指定します。サーバー・ライブラリー内のマクロをチェーニングする場合は、マク ロの名前ではなくマクロ・ファイルの名前を指定する必要があります。マクロ名は 大/小文字が区別されることに注意してください。例えば、mymacro は、myMacro、 myMACR0、MyMacro などとは異なる名前です。

呼び出し側マクロと異なる場所にあるマクロを呼び出すことはできません。具体的 には、次のとおりです。

現行セッション内のマクロは、現行セッション内にあるマクロのみを呼び出すことができます。

- パーソナル・ライブラリー内のマクロは、パーソナル・ライブラリー内にあるマ クロのみを呼び出すことができます。
- サーバー・ライブラリー内のマクロは、サーバー・ライブラリー内にあるマクロのみを呼び出すことができます。
- ユーザー・ライブラリー内のマクロは、ユーザー・ライブラリー (同じディレク トリー)内にあるマクロのみを呼び出すことができます。

「開始画面名 (Start Screen Name)」リスト・ボックスは、マクロ・ランタイムが ターゲット・マクロ内で最初に処理するマクロ画面を選択するために使用します。

- ターゲット・マクロを通常の開始画面から開始する場合は、「開始画面名 (Start Screen Name)」リスト・ボックスの *DEFAULT* 項目を選択するか、値 *DEFAULT* として評価される式を指定します。
- ターゲット・マクロを他の画面から開始する場合は、「開始画面名 (Start Screen Name)」リスト・ボックスから画面の名前を選択します。

変数の転送

「変数の転送 (Variable Transfer)」リスト・ボックスを「転送 (Transfer)」に設定 することにより (デフォルトは「転送なし (No Transfer)」)、呼び出し側マクロに 属する変数すべて (これらの変数の内容を含む) をターゲット・マクロに転送するよ うにマクロ・ランタイムに指示できます。

このように変数とその内容を転送することにより、変数を使用して呼び出し側マクロからターゲット・マクロにパラメーターを渡すことができます。

ターゲット・マクロに制御が渡された後、ターゲット・マクロは、自身が宣言した 変数の読み書きと同じように、転送変数を読み書きできます。

例えば、MacroA に 2 つの整変数 StartRow と StartCol があり、その値が 12 と 2 である場合に、MacroA が PlayMacro アクションを使用して MacroB を起動す ると、MacroB には 12 と 2 の値を持つ変数 StartRow と StartCol が最初に与え られます。

転送変数がインポート型に属していて Java オブジェクトを含んでいる場合でも、 ターゲット・マクロは転送変数を参照して Java オブジェクトのメソッドを呼び出 すことができ、またその転送変数に他のオブジェクトを書き込むこともできます。

変数を転送するための要件: ターゲット・マクロは、拡張マクロ形式を選択してい る必要があります (33 ページの『マクロ形式の選択』を参照)。

制限: 転送される変数のすべての型に、次の制限が適用されるので注意してください。

 ターゲット・マクロの「変数 (Variables)」タブの「初期値 (Initial Value)」フィ ールドで、転送変数を使用することはできません。

追加情報: ターゲット・マクロが、転送変数と同名、同型の変数を作成した場合、 マクロ・ランタイムは転送変数ではなく作成された変数を使用します。 ターゲット・マクロが型をインポートする必要がある場合: ターゲット・マクロ内 で、インポート型に属する転送変数を使用したい場合は、同じ型をターゲット・マ クロにインポートする必要はありません。型をインポートする必要がない操作の例 は、次のとおりです。

- 転送変数を属性の値として使用する。
- 転送変数に対してメソッドを呼び出す。

ただし、インポート型の名前をターゲット・マクロ内で使用したい場合は、その型 をインポートする必要があります。型をインポートする必要がある操作の例は、次 のとおりです。

- インポート型の新規変数を宣言する。
- インポート型の新規インスタンスを作成する。
- インポート型の静的メソッドを作成する。

例

次に、PlayMacro アクションの例を示します。

```
<actions>
<playmacro name="TargetMacro" startscreen="*DEFAULT*"
transfervars="Transfer" />
```

</actions>

```
図 36. PlayMacro アクションの例
```

印刷アクション (<print> エレメント)

印刷アクションを使用して、3270 ディスプレイ・セッションのセッション・ウィン ドウからテキストを印刷できます。アプリケーション画面を印刷でき、またアプリ ケーション画面からテキストの長方形領域を印刷することもできます。 3270 プリ ンター・セッションに対して使用できるものと同じプリンター・セットアップ・オ プション、およびほとんど同じページ・セットアップ・オプションを使用できま す。

印刷アクションは、3270 ディスプレイ・セッションに対してのみ使用できます。

印刷アクションは、ホスト印刷セッションを作成しません。代わりに、印刷アクシ ョンは 3270 ディスプレイ・セッション・ウィンドウに表示されたデータを印刷し ます (画面印刷)。

印刷アクションには次のものがあります。

- 印刷開始
- 印刷抽出
- 印刷終了

印刷開始アクションは、現行マクロの印刷 Bean オブジェクトのインスタンスを生成し、Bean の印刷セットアップ・オプションとページ・セットアップ・オプション

を設定します。印刷抽出アクションは、テキストを印刷 Bean に送ります。印刷終 了アクションは、印刷 Bean を終了します。

マクロ・エディターは、印刷開始、印刷抽出、および印刷終了の各アクションを別 々のタイプのアクションとして表示しますが、実際にはマクロ・オブジェクトは <print> エレメントを使用して 3 つのアクションすべてを保管します。

印刷開始

印刷開始アクションは、ユーザーが指定した印刷セットアップ・オプションとペー ジ・セットアップ・オプションを使用して、現行マクロの印刷 Bean オブジェクト のインスタンスを生成します。

印刷開始アクションを実行する前に、マクロ・ランタイムは現行マクロの印刷 Bean がすでに存在するかどうか確認します。存在する場合、マクロ・ランタイムは既存 の印刷 Bean を終了し、印刷開始アクションを実行して新規印刷 Bean のインスタ ンスを生成します。

プリンター・セットアップとページ・セットアップ:新規印刷 Bean のプリンタ ー・セットアップ・オプションを設定するには、「プリンター・セットアップ (Printer Setup)」をクリックします。制御できるプリンター・セットアップ・オプシ ョンは、3270 プリンター・セッションに対して使用できるものと同じです。これら のオプションには、プリンター出力先 (Windows プリンター、その他のプリンタ ー、またはファイル)、プリンター定義テーブル、ファイル出力先の Adobe PDF 出 力などがあります。

新規印刷 Bean のページ・セットアップ・オプションを設定するには、「ページ・ セットアップ (Page Setup)」をクリックします。制御できるページ・セットアッ プ・オプションは、3270 プリンター・セッションに対して使用でき、また 3270 表 示データ・ストリーム (LU2) にも適用できるものと同じです。これらのオプション には、フォント、ヌル (0x00) の処理、プリンター・フォントのコード・ページなど があります。

現行マクロの印刷 Bean に対して指定したプリンター・セットアップ・オプション とページ・セットアップ・オプションは、以下に対するプリンター・セットアッ プ・オプションとページ・セットアップ・オプションには影響しません。

- 別の 3270 ディスプレイ・セッション上で実行されるマクロの印刷 Bean。
- 3270 ディスプレイ・セッション内での ZipPrint。
- 3270 ディスプレイ・セッション内での「ファイル」 > 「画面印刷 (Print Screen)」操作。
- 3270 プリンター・セッション。

ただし、印刷出力先が Windows プリンターで、Microsoft Windows のプリンタ ー・セットアップ・ダイアログでその Windows プリンターの構成を変更した (横 長方向から縦長方向への変更など) 場合は、その Windows プリンターを使用する Host On-Demand 印刷アクティビティーすべてに、その構成変更の影響が及びま す。これには以下も含まれます。

- 3270 ディスプレイ・セッション上で実行されるマクロの印刷 Bean。
- 3270 ディスプレイ・セッション内での ZipPrint。

3270 印刷セッションからの印刷。

変数への戻りコードの割り当て: 印刷開始アクションが正常に行われたことを確認 するには、「変数に戻りコードを割り当てる (Assign Return Code to a Variable)」をクリックし、印刷開始アクションからの戻りコードを格納する変数を 選択します。

注: 戻りコードは、正常な場合の 0、あるいは、失敗した場合の -1 のいずれかで す。また、プリンター・エラーで、マクロがエラーで終了する場合があります。

印刷抽出

印刷抽出アクションは、指定した 3270 ディスプレイ・セッション・ウィンドウの 長方形領域からテキストをコピーし、現行印刷 Bean を使用してそのテキストを印 刷します。

印刷抽出アクションを実行する前に、マクロ・ランタイムは現行マクロの印刷 Bean がすでに開始されているかどうか確認します。開始されていない場合、マクロ・ラ ンタイムは、デフォルト・プリンター・セットアップ・オプションとデフォルト・ ページ・セットアップ・オプションを指定して印刷開始アクションを実行し、その 後で印刷抽出アクションを実行します。

印刷する領域の指定: 印刷するセッション・ウィンドウの領域を指定するには、マ ーキング長方形を使用して行と列の座標を取り込むことができます。また別の方法 として、「抽出アクション (Extract action)」ウィンドウの「行 (Row)」フィールド と「列 (Column)」フィールドに、テキスト域の行と列の座標を入力することもでき ます。

マーキング長方形を使用する場合(203 ページの『マーキング長方形の使用』を参照)、マクロ・ランタイムは、マーキング長方形の左上隅の行と列の座標を1 組目の行と列の値(「抽出アクション(Extract action)」ウィンドウの「上隅(Top Corner)」)に書き込み、右下隅の行と列の座標を2 組目の行と列の値(「下隅(Bottom Corner)」)に書き込みます。

行と列の値をユーザー自身が入力する場合は、1 組目の行と列の座標を 1 組目の行 と列の値(「抽出アクション (Extract action)」ウィンドウの「上隅 (Top Corner)」)に入力し、2 組目の座標のセットを 2 組目の行と列の値(「下隅 (Bottom Corner)」)に入力します。必要な座標を判別するための補助として、セッ ション・ウィンドウでテキスト・カーソルを使用できます(203 ページの『セッシ ョン・ウィンドウのテキスト・カーソルの使用』を参照)。

「行 (下隅) (Row (Bottom Corner))」入力フィールドに -1 を入力すると、セッシ ョン・ウィンドウのデータ域の最終行を示すことができます。この機能は、ユーザ ーが高さの異なる (25、43、50 など) セッション・ウィンドウを使用していて、最 終行までデータをキャプチャーしたい場合に便利です。同様に、「列 (下隅) (Column (Bottom Corner))」入力フィールドに -1 を入力すると、セッション・ウ ィンドウのデータの最終列を示すことができます (41 ページの『行または列の負 の値の意味』を参照)。 変数への戻りコードの割り当て: 印刷抽出アクションが正常に行われたことを確認 するには、「変数に戻りコードを割り当てる (Assign Return Code to a Variable)」をクリックし、印刷抽出アクションからの戻りコードを格納する変数を 選択します。

注: 戻りコードは、正常な場合の 0、あるいは、失敗した場合の -1 のいずれかで す。また、プリンター・エラーで、マクロがエラーで終了する場合があります。

セッションの指定: 「ホスト ID (Host ID)」フィールドを使用して、このアクショ ンを実行するセッションを指定します。

- マクロを起動するセッションでこのアクションを実行する場合は、「ホスト ID (Host ID)」フィールドをブランクのままにする。
- 別のセッションでこのアクションを実行する場合は、「ホスト ID (Host ID)」 フィールドにセッション参照を入力する(196 ページの『ホスト ID の指定』を 参照)。

参照されたセッションがアクティブである場合は、このセッションに任意の自動編 集機能を使用できます (197 ページの『異なるセッションでの自動編集機能の使 用』を参照)。

印刷終了

印刷終了アクションは、現行印刷 Bean が存在すれば、その Bean を終了します。 現行印刷 Bean が存在しない場合、このアクションの効果はありません。

変数への戻りコードの割り当て: 印刷終了アクションが正常に行われたことを確認 するには、「変数に戻りコードを割り当てる (Assign Return Code to a Variable)」をクリックし、印刷終了アクションからの戻りコードを格納する変数を 選択します。

注: 戻りコードは、正常な場合の 0、あるいは、失敗した場合の -1 のいずれかで す。また、プリンター・エラーで、マクロがエラーで終了する場合があります。

プロンプト・アクション (<prompt> エレメント)

プロンプト・アクションは、ユーザーによる直接のキーボード入力を 3270 または 5250 アプリケーションに送るか、変数に格納するための強力な機能を備えていま す。

プロンプト・アクションは、セッション・ウィンドウの前面にプロンプト・ウィン ドウを表示します。このウィンドウには、メッセージ、入力フィールド、および 3 つのボタン (「OK」、「取り消し (Cancel)」、「ヘルプ」) があります。ユーザー が入力フィールドにテキストを入力して「OK」をクリックした後、プロンプト・ア クションはその入力データを使用して、次のどちらかまたは両方を行います。

- プロンプト・アクションは、セッション・ウィンドウの入力フィールドに入力デ ータを入力する。
- プロンプト・アクションは、入力データをストリングとして解釈し、入力データ を変数に格納する。

このアクションの代表的な用途は (ただしこの用途だけではありません)、ユーザー によるパスワードの入力を可能にすることです。マクロがホストにログオンした り、アクセスにパスワードを必要とするアプリケーションを開始したりしなければ ならないシナリオはよくあります。パスワードは機密データであり、また通常はと きどき変更されるため、パスワードを即時値としてマクロにコーディングすること は適切でないことがよくあります。

プロンプト・アクションを使用して、ユーザーにパスワードの入力を促し、入力フ ィールドにパスワードを入力するようにユーザーに指示するメッセージを表示でき ます。ユーザーが「OK」をクリックした後、マクロ・ランタイムは、セッション・ ウィンドウ内の指定した行と列の位置に入力データを入力します。入力シーケンス には、[enterreset] などのアクション・キーを含めることができるので、ユーザーが MyPassword[enterreset] と入力した場合、マクロ・ランタイムはパスワードをパ スワード・フィールドに入力できるだけでなく、ログオンまたはアクセスのアクシ ョンを完了するキーを入力することもできます。(また、プロンプト・アクション の直後の入力アクションにアクション・キーを組み入れることもできます。)

プロンプト・ウィンドウの表示

プロンプト・ウィンドウの各部分: プロンプト・テキスト ('Please type your password:' など) は、「プロンプト・テキスト (Prompt Text)」フィールドではな く「プロンプト名 (Prompt Name)」フィールドに入力する必要があります (「プロ ンプト・テキスト (Prompt Text)」フィールドは、特定のプロンプト・アクション の詳細を説明するメモの保管に使用できるオプションのフィールドです)。

マクロ・ランタイムが表示するプロンプト・ウィンドウには、次の特性があります。

- プロンプト・ウィンドウはセッション・ウィンドウの前面に表示され、システムのデスクトップ・ウィンドウの中央に配置されます。
- プロンプト・ウィンドウのタイトルは、「プロンプト・タイトル」フィールドの 値になります。ただし、フィールドがブランクであるか、マクロの中に複数のプ ロンプトがあり、マクロの開始時にすべてのプロンプトを表示するように構成さ れている場合を除きます。この2つの場合は、プロンプト・ウィンドウのタイ トルは、「プロンプト (Prompt)」となります。
- 「プロンプト名」フィールドに入力したメッセージは、プロンプト・ウィンドウの中央に表示され、その後に入力フィールドが表示されます。
- プロンプト・ウィンドウの下部にあるボタン行には、次の3つのボタンがあります。
 - 「OK」ボタンが押されると、マクロ・ランタイムは入力フィールドの内容を 処理します。
 - 「取り消し (Cancel)」ボタンは、マクロを停止します。
 - 「ヘルプ (Help)」ボタンは、プロンプト・ウィンドウの使用法を説明するヘ ルプ・テキストを表示します。

デフォルト応答: 「デフォルト応答 (Default Response)」フィールド (オプション) には、プロンプト・ウィンドウが表示されたときに、プロンプト・ウィンドウの入 力フィールドに表示したいデフォルト応答のテキストを入力できます。ユーザーが プロンプト・ウィンドウの入力フィールドにキーボードで入力せず、単に「OK」を クリックして入力完了を指示した場合、マクロ・ランタイムは入力フィールドに入 っているデフォルト応答を処理します。 例えば、ユーザーが通常は ApplicationA を使用し、ときどき ApplicationB を使 用する場合は、「デフォルト応答 (Default Response)」フィールドに ApplicationA と入力できます。マクロ・ランタイムがプロンプト・アクションを実行すると、入 カフィールドにテキスト ApplicationA がすでに表示された状態で、プロンプト・ ウィンドウが表示されます。ユーザーは「OK」をクリックするか (この場合、マク ロは入力フィールドの内容として ApplicationA を処理する)、入力フィールドに ApplicationB と入力して「OK」をクリックします (この場合、マクロは入力フィ ールドの内容として ApplicationB を処理する)。

パスワード応答: 「パスワードの応答」リスト・ボックスで true を選択する場合 (デフォルトは false)、ユーザーがプロンプト・ウィンドウの入力フィールドにキー 入力するたびに、マクロ・ランタイムはそのキーに関連した文字の代わりにアスタ リスク (*) を表示します。

例えば、「パスワードの応答」リスト・ボックスを true に設定すると (または実行時に true に解決されると)、ユーザーが 'Romeo' と入力した場合に、マクロ・ラン タイムは入力フィールドに ***** を表示します。

応答の要求: 「応答の要求 (Require Response)」リスト・ボックスで true を選択 する場合 (デフォルトは false)、次のようになります。

- マクロ・ランタイムは、入力フィールドの右にテキスト・ストリング(必要 (required))と表示して、この入力フィールドへの入力が必要であることをエン ド・ユーザーに示す。
- マクロ・ランタイムは、プロンプト・ウィンドウの入力フィールドにテキストが 入力されるまで「OK」ボタンを使用不可にする。
 - 入力フィールドにテキストを入力するには、デフォルトの応答を指定する
 か、入力フィールドにテキストを入力する。
 - 「OK」が使用可能の場合、エンド・ユーザーは通常どおり「OK」または 「取り消し (Cancel)」をクリック可能。
 - 「OK」をクリックすると、マクロ・ランタイムがプロンプト・アクション を処理するようになり、マクロの処理が継続する。
 - 「取り消し (Cancel)」をクリックすると、マクロ再生が終了する。
 - 「OK」が使用可能になっていない場合、エンド・ユーザーは「取り消し (Cancel)」をクリック可能。
 - 「取り消し (Cancel)」をクリックすると、マクロ再生が終了する。

したがって、「応答の要求 (Require Response)」を true に設定すると、先に進む 前に応答が必要であることをエンド・ユーザーに気付かせ (入力フィールドの右に 「(必要 (required))」を表示する)、「OK」をクリックする前に入力フィールドに テキストを入力するようエンド・ユーザーに要求することができます (入力フィー ルドにテキストが入力されるまで「OK」を使用不可にする)。ただし、プロンプ ト・アクションにデフォルトの応答が含まれる場合は、「OK」が使用可能になり、 デフォルトの応答が入力フィールドに表示されます。

「応答の要求 (Require Response)」リスト・ボックスで false を選択する場合、次のようになります。

 マクロ・ランタイムは、入力フィールドの右にテキスト・ストリング「(必要 (required))」と表示しない。

- マクロ・ランタイムは、入力フィールドにテキストが入力されているかどうかに
 関係なく、プロンプト・ウィンドウが表示されると即座にプロンプト・ウィンドウの「OK」ボタンを使用可能にする。
 - ユーザーは通常どおり「OK」または「取り消し (Cancel)」をクリックできる。
 - 「OK」をクリックすると、マクロ・ランタイムがプロンプト・アクション を処理するようになり、マクロの処理が継続する。プロンプト・アクショ ンでは、入力フィールドがブランクの場合、マクロ・ランタイムは入力キ ー・シーケンスをセッション・ウィンドウに送信しません。
 - 「取り消し (Cancel)」をクリックすると、マクロ再生が終了する。

したがって、「応答の要求 (Require Response)」を false に設定すると、プロンプトの入力フィールドがブランクの場合でも、ユーザーが「OK」をクリックしてマクロの処理を継続できるようになります。

<HAScript> エレメント (または <actions> エレメント)の promptall 属性が true に設定されている場合で、マクロ (またはマクロ画面) に複数のプロンプト・アクションがあり、「応答の要求 (Require Response)」を true に設定してある場合、マクロ再生の開始時 (またはマクロ画面再生の開始時) にマクロ・ランタイムが単一のプロンプト・ウィンドウにすべてのプロンプト入力フィールドを表示すると、すべての必要入力フィールドにテキストが入力されるまで、マクロ・ランタイムは「OK」ボタンを使用可能にしません (118 ページの『promptall 属性』を参照)。

入力フィールドの内容の処理

応答の長さ: 「応答の長さ (Response Length)」フィールドの値は、入力フィール ドのサイズではなく、マクロ・ランタイムがユーザーに許可する入力フィールドへ の入力文字数を指定します。

例えば、「応答の長さ (Response Length)」フィールドを 10 に設定すると、マク ロ・ランタイムは、入力フィールドへの入力を 10 文字のみユーザーに許可しま す。

アクション・キーとホスト・アクション・キーの変換: マクロ作成者(「デフォル ト応答 (Default Response)」入力フィールド)とユーザー(「プロンプト (Prompt)」ウィンドウの入力フィールド)の両方が、入力アクションの「ストリン グ (String)」フィールドの場合と同様に(98ページの『入力ストリング』を参 照)、アクション・キー ([enterreset]、[copy] など)を使用できます。

「ホスト・アクション・キーの変換 (Translate Host Action Keys)」リスト・ボッ クスとその効果は、入力アクションの「ホスト・アクション・キーの変換 (Translate Host Action Keys)」リスト・ボックスとまったく同じです (99 ページ の『ホスト・アクション・キーの変換』を参照)。このリスト・ボックスを true (デ フォルト値) に設定すると、マクロ・ランタイムはアクション・キー・ストリング ([copy] など) をリテラル・ストリングとしてではなくアクション・キーとして解釈 します。

セッション・ウィンドウでの入力シーケンスの処理

「行 (Row)」フィールドと「列 (Column)」フィールドを使用して、マクロ・ラン タイムによる入力シーケンスの入力を開始するセッション・ウィンドウの行と列の 位置を指定します。マクロ・ランタイムがテキスト・カーソルの現在位置から入力 シーケンスの入力を開始するようにするには、「行 (Row)」フィールドと「列 (Column)」フィールドのどちらかまたは両方を 0 に設定します。入力アクションと 同様に、行と列の位置は 3270 または 5250 入力フィールドの中にあることが必要 で、そうでなければセッション・ウィンドウは入力を禁止してエラー・シンボルを オペレーター情報域に表示します。この応答は、実際のユーザーからのキーボード 入力に対する応答とまったく同じです。

「ホスト・フィールドの消去 (Clear Host Field)」リスト・ボックスを true に設定 すると、マクロ・ランタイムは入力開始前に入力フィールドの内容を消去します。

「カーソルを入力の最後に移動 (Move Cursor to End of Input)」フィールドに は、入力アクションの同名のボタンと同じ機能と効果があります (99 ページの 『カーソルを入力の最後に移動』を参照)。

「画面に書き込まない (Don't Write to Screen)」リスト・ボックスを true に設定 すると、入力フィールドに入力シーケンスを表示しないようにマクロ・ランタイム に指示できます。このフィールドは、「変数に割り当てる (Assign to a Variable)」チェック・ボックスが選択されている場合にのみ使用可能になります。

変数への入力シーケンスの割り当て

「変数に割り当てる (Assign to a Variable)」チェック・ボックスにチェックマー クを付けると、入力シーケンスを変数に格納するようにマクロ・ランタイムに指示 できます。

新規変数を作成するには、リスト・ボックスで「<新規変数> (<New Variable>)」項 目をクリックします。新規変数を指定するためのポップアップ・ウィンドウで、現 行マクロが別のマクロから継承する変数の名前を指定し、また現行マクロ内で作成 する新規変数の名前を指定することもできます。現行マクロ内で新規変数を作成す る場合は、「このマクロで変数を作成 (Create variable in this macro)」チェッ ク・ボックスを選択し、新規変数の型を選択します。

マクロ・ランタイムは入力シーケンスをストリングとして保管するので、入力を受け取る変数としてストリング変数を指定できます。ただし、変数がストリング以外の型である場合、マクロ・ランタイムは通常の規則に従って、入力データをターゲット変数のデータ型に変換することを試みます(40ページの『自動データ型変換』を参照)。

promptall 属性

すべての <prompt> エレメントのポップアップ・ウィンドウを 1 つの大きなプロ ンプト・ウィンドウに結合し、マクロの再生開始時にこの大きなプロンプト・ウィ ンドウを表示するように、マクロ・ランタイムに指示できます。このためには、 <HAScript> エレメントの promptall 属性を true に設定します (234 ページの 『<HAScript> エレメント』を参照)。

<actions> エレメントの promptall 属性も同様に機能します (220 ページの 『<actions> エレメント』を参照)。 セッションの指定

「ホスト ID (Host ID)」フィールドを使用して、このアクションを実行するセッションを指定します。

- マクロを起動するセッションでこのアクションを実行する場合は、「ホスト ID (Host ID)」フィールドをブランクのままにする。
- 別のセッションでこのアクションを実行する場合は、「ホスト ID (Host ID)」 フィールドにセッション参照を入力する(196 ページの『ホスト ID の指定』を 参照)。

参照されたセッションがアクティブである場合は、このセッションに任意の自動編 集機能を使用できます (197 ページの『異なるセッションでの自動編集機能の使 用』を参照)。

プログラム実行アクション (<runprogram> エレメント)

プログラム実行アクションは、ネイティブ・アプリケーションを起動し、オプショ ンでその終了を待ちます。アプリケーションの入力パラメーターを指定でき、変数 に戻りコードを格納できます。

システム・ランタイムによって実行できる任意のアプリケーションを起動できま す。

プログラム実行アクションには、次のようにさまざまな用途があります。

- マクロに必要なデータを準備する、またはマクロが準備したデータを使用するネ イティブ・アプリケーションを起動する。
- マクロが開始しようとしているアクションのためにワークステーションを準備する(例えば、ネットワーク接続を行う)ネイティブ・アプリケーションを起動する。
- システム状態の状況を検出し、状況をマクロに報告するネイティブ・アプリケーションを起動する。

ネイティブ・アプリケーションの起動

ネイティブ・アプリケーションを起動するファイルの完全なパスと名前を「プログ ラム (Program)」入力フィールドに指定する必要があります。例を次に示します。 'c:¥¥Program Files¥¥MyApp¥¥bin¥¥myapp.exe'

上の例の中で、単一の円記号 (¥) が 2 つの円記号 (¥¥) によって表されている点に 注意してください。この理由は、拡張マクロ形式では円記号が特殊記号であるた め、円記号 + 文字自体として表現する必要があることです (34 ページの『拡張マ クロ形式のストリング表記規則』を参照)。

「パラメーター (Parameters)」フィールドには、ネイティブ・アプリケーションに 渡す必要があるパラメーターすべてを指定します。

ネイティブ・アプリケーションの終了の待機

ネイティブ・アプリケーションが終了するまで待機するようにマクロ・ランタイム に指示する場合は、「プログラムの待機 (Wait for Program)」リスト・ボックスを true に設定します。デフォルトは false (マクロ・ランタイムは待機しない) です。

戻りコードの取り込み

「終了コードを変数に割り当てる (Assign Exit Code to Variable)」チェック・ボ ックスを選択し、変数名を指定することによって、ネイティブ・アプリケーション から戻された状況コードを変数に取り込むことができます。

ネイティブ・アプリケーションの起動例

次の例では、ネイティブ・アプリケーションを起動し、終了するまで待機し、アプ リケーションからの戻りコードをメッセージ・ウィンドウに表示します。この例で は、プログラム実行アクション、メッセージ・アクションのアクション・エレメン トを使用します。

このマクロ・スクリプトのテキストを本書からシステム・クリップボードにコピー し、システム・クリップボードからコード・エディターにコピーできます (204 ペ ージの『本書からコード・エディターへのスクリプトのコピー・アンド・ペース ト』を参照)。このスクリプトをマクロ・エディターに保管した後、マクロ・エディ ターまたはコード・エディターを使用してスクリプトを編集できます。

この例については、次の点に注意してください。

- この例は、RUN PROGRAM という名前の 1 つのマクロ・スクリプト全体で構成されます。
- <actions> エレメント内では次のアクションが行われます。
 - <runprogram> エレメントがネイティブ・アプリケーションを起動し、アプリケーションが戻るまで待機し、戻りコードを \$intReturn\$ に格納する。
 - メッセージ・アクションが、\$intReturn\$の値をメッセージ・ウィンドウに表示する。

```
<HAScript name="g1" description=" " timeout="60000" pausetime="300"</pre>
            promptall="true" author="" creationdate="" supressclearevents="false"
            usevars="true" ignorepauseforenhancedtn="false"
            delayifnotenhancedtn="0">
  <vars>
      <create name="$intReturn$" type="integer" value="0" />
   </vars>
   <screen name="Screen1" entryscreen="true" exitscreen="true" transient="false">
      <description>
         <oia status="NOTINHIBITED" optional="false" invertmatch="false" />
      </description>
      <actions>
         <runprogram exe=
                   "%ProgramFiles%¥Windows NT¥Accessories¥wordpad.exe"
                   param="'c:\\text{tm\\text{m}\\text{mw} file.doc'" wait="true"
                  assignexitvalue="$intReturn$" />
         <message title="" value="'Return value is '+$intReturn$" />
      </actions>
      <nextscreens timeout="0" >
      </nextscreens>
   </screen>
</HAScript>
```

図 37. サンプル・コード RUN PROGRAM

SQLQuery アクション (<sqlquery> エレメント)

SQLQuery アクションは、非常に便利で強力なアクションです。このアクションに より、SQL ステートメントをホスト・データベースに送信し、SQL ステートメン トで生成されるデータを検索し、データをグローバル変数に書き込むか、データを ファイルに書き込むか、データを表示することができます。(対になるアクション FileUpload では、ファイルのアップロード・コマンドをホスト・データベースに送 信できます。 93 ページの『FileUpload アクション (<fileupload> エレメント)』 を参照してください。)

SQLQuery アクションは、マクロをサポートする任意のタイプの Host On-Demand セッションで使用できます (3270 ディスプレイ、5250 ディスプレ イ、VT ディスプレイ、または CICS ゲートウェイ)。

接続先のデータベース・サーバーは、エミュレーター・セッションを実行している ホストとは別のホストにあってもかまいません。

SQL ステートメントを手動で作成し、SQL ウィザードを使用して SQL ステートメ ントを構成およびテストし、現行セッションまたは SQL ステートメントのライブ ラリーから SQL ステートメントをインポートできます。

SQLQuery アクションでは、タイプが Select の SQL ステートメントのみサポート されています。タイプが Insert、Update、または Delete の SQL ステートメント はサポートされていません。

2 つのセクション: ステートメントと結果

SQLQuery アクション・ウィンドウには、2 つの主なセクション (ステートメント・セクションと結果セクション) があります。

ステートメント・セクションはウィンドウの上部にあり、「データベース URL (Database URL)」、「ドライバー ID (Driver Identifier)」、「ドライバー・クラス (Driver Class)」、「ユーザー ID (User ID)」、「パスワード (Password)」、およ び「ステートメント (Statement)」フィールドが含まれています。このセクションの 情報は、次の 3 つの方法で変更できます。

- SQL ウィザードで SQL ステートメントを作成する。
- SQL ステートメントをインポートする。
- フィールドに情報を入力する。

これらのフィールドは、入力することによっていつでも編集することができます。

結果セクションはウィンドウの下部にあり、「結果の出力先 (Output Result To)」、「ファイル名」、「Web ブラウザーで表示 (Show in Web browser)」、 「ファイル・タイプ」、「出力ダイアログで待機 (Hold on output dialog)」、 「上書き (Overwrite)」、および「付加 (Append)」フィールドが含まれています。

SQL ウィザードの使用

SQL ウィザードを使用して、SQL ステートメントを作成しテストできます。マク ロ・エディターの「ステートメント (Statement)」に SQL ステートメントのテキス ト全体を入力する場合と比較して、SQL ウィザードのグラフィカル・ユーザー・イ ンターフェースを使用すると、SQL ステートメントをより簡単に作成できます。また、SQL ウィザードでは、作業中の SQL ステートメントを実行して結果を表示することができます。

- 1. 「SQL ウィザード (SQL Wizard)」をクリックして、SQL ウィザードを開始する。
 - ステートメント・セクションのフィールドにすでに情報が含まれている場合、Host On-Demand はこの情報を使用して、SQL ウィザードの対応するフィールドを初期化する。
- 2. SQL ウィザードを使用して、SQL ステートメントを作成しテストする。
- 3. マクロを変更しないで SQL ウィザードを閉じるには、「取り消し (Cancel)」 をクリックする。
- 4. SQL ステートメントをマクロに保管するには、次のいずれかのアクションを実 行する。
 - SQL ウィザードの「検討 (Review)」タブで「保管」をクリックする。
 - SQL ウィザードの「結果 (Results)」タブで「SQL を保管 (Save SQL)」を クリックする。

マクロ・エディターは、SQL ウィザードで作成した情報をステートメント・セ クションの適切なフィールドに書き込みます。ステートメント・セクションのフ ィールドにすでに情報が含まれている場合は上書きされます。マクロ・エディタ ーがフィールドに書き込む情報がストリングの場合 (例えば、「データベース URL (Database URL)」フィールドに書き込まれる情報)、マクロ・エディター は基礎マクロ・タイプに基づいて自動的にストリングを正しくフォーマット設定 します (33 ページの『基本マクロ形式と拡張マクロ形式の比較』を参照)。次 のフィールドが更新されます。

- ステートメント・セクションのフィールド:
 - データベース URL (Database URL)
 - ドライバー ID (Driver Identifier)
 - ドライバー・クラス (Driver Class)
 - ユーザー ID (User ID)
 - パスワード (Password)
 - ステートメント (Statement)
- 結果セクションのフィールド:
 - SQL ウィザードの「出力 (Output)」タブで照会結果をモニターに出力す るよう選択すると、マクロ・エディターは次のようにします。
 - 「結果の出力先 (Output Result To)」フィールドを「ダイアログ (Dialog)」に設定する。
 - SQL ウィザードの「出力 (Output)」タブで照会結果をファイルに出力す るよう選択すると、マクロ・エディターは次のようにします。
 - 「結果の出力先 (Output Result To)」フィールドを「ファイル」に設 定する。
 - 「ファイル名」フィールドを SQL ウィザードで指定したファイル名 に設定する。

- 「ファイル・タイプ」フィールドを SQL ウィザードで指定したファ イル・タイプに設定する。
- SQL ウィザードで上書きまたは付加のどちらを選択したかに応じて、 「上書き (Overwrite)」ラジオ・ボタンまたは「付加 (Append)」ラジ オ・ボタンを選択する。
- ファイル・タイプが HTML または XML の場合、HTML 設定または XML 設定を保管する。これらの設定はマクロ・エディターでは表示さ れませんが、コード・エディターを使用すると、<sqlquery> エレメン トの mlprops 属性に表示されます (252 ページの『<sqlquery> エレ メント』を参照)。
- 5. 「取り消し (Cancel)」をクリックして、SQL ウィザードを閉じる。

SQL ステートメントのインポート

「照会のインポート (Import Query)」をクリックして、SQL ステートメントをイ ンポートします。現行セッションまたはクライアント・ワークステーションの個人 用ライブラリーから SQL ステートメントをインポートできます。

後でインポートできるように SQL ステートメントを保管するには、SQL ウィザー ドを開始し、SQL ステートメントを作成またはオープンして SQL ステートメント を実行し、「結果 (Results)」タブに移動して「SQL を保管 (Save SQL)」をクリッ クします。SQL ステートメントを現行セッションまたは個人用ライブラリーに保管 するには、「保管されたステートメント (Saved statements)」ダイアログを使用し ます。

SQL ステートメントをインポートすると、マクロ・エディターはインポートされた SQL ステートメントの内容をステートメント・セクションのフィールドに書き込み ます。また、マクロ・エディターは、新規情報を基礎マクロ・タイプに合わせて自 動的に正しくフォーマット設定します (33 ページの『基本マクロ形式と拡張マク ロ形式の比較』を参照)。ステートメント・セクションのフィールドにすでに情報が 含まれている場合は上書きされます。

ステートメント・セクションのフィールドの使用

SQL ウィザードを使用して SQL ステートメントを作成したり、SQL ステートメン トをインポートする代わりに、ステートメント・セクションのフィールドに情報を 直接入力することができます。SQL ステートメントの作成またはインポート後に、 どのフィールドにも入力できます。

データベースの URL: 「データベース URL (Database URL)」フィールドに、デ ータベースへのアクセスを提供するデータベース・サーバーの URL を入力しま す。データベース URL の形式は、データベースへのアクセスに使用する Java Database Connectivity (JDBC) ドライバーのタイプによって異なります (ドライバ ーについて詳しくは、 124 ページの『ドライバー ID とドライバー・クラス』を参 照)。 124 ページの表 18 は、Host On-Demand に組み込まれているドライバーの データベース URL を示しています。

表 18. データベース URL の形式

Host On-Demand で 提供されるドライバー:	データベース URL の形式:	例えば、次のとおりです。
AS/400 Toolbox for Java	jdbc:as400://[host]	• jdbc:as400://myISeries
		• jdbc:as400://9.88.24.163

実際のデータベース URL (jdbc:as400://myISeries など) では、上記の形式例に示 されている大括弧は使用しない でください。

リモート・サーバーは、Host On-Demand エミュレーター・セッションの接続先の ホストとは別のホストに配置することができます。例えば、SQLQuery アクション は iSeries ホストを指定できます。これは、同じ SQLQuery アクションが、zSeries ホストに接続されている 3270 ディスプレイ・セッションで実行中のマクロの一部 であっても同様です。

Host On-Demand で提供されるドライバー以外の JDBC ドライバーを使用する場合、データベース URL の正しい形式については、ドライバー・ベンダーの提供する資料を調べてください。

ドライバー ID とドライバー・クラス: SQLQuery アクションがデータベースにア クセスするために使用する JDBC ドライバーは、Java クライアント・パッケージ です。これは、リモート・ホスト上のサーバー・プログラムと通信する Host On-Demand クライアント・ワークステーション上にあります。リモート・ホスト 上のこのサーバー・プログラムを使用すると、データベースへのアクセスを行うこ とができます。

Host On-Demand クライアントには、AS/400 Toolbox for Java の JDBC ドライ バーがすでに組み込まれています。このドライバーは、エミュレーター・クライア ントの一部として自動的にダウンロードされます。このドライバーにより、クライ アントは、正しく構成された iSeries または AS/400 上の、DB2/400 データにアク セスできます。

別の JDBC ドライバーが必要な場合は、リモート・データベース・サーバーの管理 者と連絡を取ってドライバーを取得してください。ドライバーを Host On-Demand クライアント・ワークステーションにデプロイするには、いくつかのアクションを 実行する必要があります。

マクロ・エディターの「SQLQuery アクション (SQLQuery action)」ウィンドウの 「ドライバー ID (Driver Identifier)」リスト・ボックスで Host On-Demand で提 供されるドライバーを選択するか、別のドライバーを使用する場合には「その他 (0ther)」を選択します。

「ドライバー・クラス (Driver Class)」フィールドには、ドライバーの完全修飾 Java クラス名が入ります。Host On-Demand で提供されるドライバーを選択する 場合、マクロ・エディターは「ドライバー・クラス (Driver Class)」フィールドに クラス名 (com.ibm.as400.access.AS400JDBCDRIVER) を表示します。このクラス名を 変更することはできません。一方、「ドライバー ID (Driver Identifier)」リスト・ ボックスで「その他 (Other)」を選択する場合は、「ドライバー・クラス (Driver Class)」フィールドにドライバーの完全修飾クラス名を入力する必要があります。完 全修飾クラス名が分からない場合は、ドライバーのプロバイダーにお問い合わせく ださい。名前を入力する場合は、大/小文字に注意してください (例えば、com と COM は異なります)。

「ドライバー ID (Driver Identifier)」リスト・ボックスの選択可能なドライバーの リストにドライバーを追加する場合 (毎回「その他 (Other)」を選択してクラス名 を入力しないようにする場合) は、ドライバーを Host On-Demand に登録できま す (オンライン・ヘルプの『JDBC ドライバーの登録 (Registering a JDBC driver)』を参照)。

ユーザー ID とパスワード: データベース接続でユーザー ID とパスワードが必要 な場合は、「ユーザー ID (User ID)」フィールドにユーザー ID、「パスワード (Password)」フィールドにパスワードを入力します。

Host On-Demand は、「パスワード (Password)」フィールドに入力するキー・シ ーケンスを暗号化します。この暗号化は、入力アクションで「パスワード (Password)」チェック・ボックスを選択する場合に使用される暗号化とまったく同 じ方法で機能します (99 ページの『パスワード』を参照)。要確認:

- 「パスワード (Password)」フィールドにパスワード (mypass など) を入力する 場合、マクロ・エディターはパスワードをアスタリスク (******) で表示する。
- 入力フォーカスを別の入力フィールドに移動すると、マクロ・エディターでは次のようになる。
 - 1. 暗号化されたパスワードが生成される (q0eqOskTUBQ= など)。
 - 2. 「パスワード (Password)」フィールドに、アスタリスクを使用した暗号化さ れたパスワード (**********) が表示される。(暗号化されたパスワードの 実際の文字は、コード・エディターで確認できます。)
- パスワードはストリングである。したがって、拡張マクロ形式を使用する場合は、パスワードを単一引用符で囲んで入力してください(例えば、'mypass')。マクロ・エディターは、単一引用符を含めたストリング全体を暗号化します。
- マクロ・エディターによってパスワードが暗号化された後でパスワードを変更する必要がある場合は、新規パスワードを入力する前に、フィールドのすべての文字を完全に削除する必要がある。

ステートメント: 「ステートメント (Statement)」フィールドに、SQL ステートメ ントを入力するか貼り付けます。「ステートメント (Statement)」フィールドに SQL ステートメントがすでに含まれている場合は、(SQL ウィザードを使用して SQL ステートメントがインポートまたは作成されている場合でも) 編集できます。

マクロ・エディターは、SQL ステートメントの形式の妥当性をチェックしません。 形式が無効の場合、マクロ・ランタイムが SQLQuery アクションを処理するときに ランタイム・エラーが発生します。

SQL ステートメントに精通していない場合は、Database On-Demand セッション でステートメントを作成およびテストしてから、そのステートメントを「ステート メント (Statement)」フィールドにコピーして貼り付けると良いでしょう。これによ り、SQL ステートメントの構文および内容が正しいものになります。

SQL ステートメントは、複数行にまたがって書き込むことも (Database On-Demand セッションの「検討 (Review)」タブに表示される場合のように)、1

行に書き込むこともできます。図 38 および図 39 は、複数行にまたがって書き込 まれた SQL ステートメントと、1 行に書き込まれた SQL ステートメントを示し ています。どちらも正しい方法です。

図 38. 複数行にまたがって書き込まれた SQL ステートメント

SELECT * FROM HODTEST.EX01 WHERE((HODTESET.EX01.DESCRIPT is not null))

図 39.1 行に書き込まれた同じ SQL ステートメント

拡張マクロ形式を使用する場合、SQL ステートメントを単一引用符で囲み、特殊文 字の規則に従う必要があります (33 ページの『基本マクロ形式と拡張マクロ形式 の比較』を参照)。下記の図 40 および 図 41 は、基本マクロ形式用と拡張マクロ 形式用の同じ SQL ステートメントを示しています。

```
select * from hodtest.ex01 where
        ((hodtest.ex01.descript='Edit Products'))
```

図 40. 基本マクロ形式用の SQL ステートメント

'select * from hodtest.ex01 where ((hodtest.ex01.descript=¥'Edit Products¥'))'

図 41. 拡張マクロ形式用の同じ SQL ステートメント

予約語 (select など) や、データベース名およびフィールド (hodtest.ex01.descript など) には大文字でも小文字でも使用できますが、マッチ ング・ストリング ('Edit Products' など) には大/小文字をその通りに使用する必 要があります。したがって、 127 ページの図 42 の 2 つの SQL ステートメント は同等です。 select * from hodtest.ex01 where ((hodtest.ex01.descript='Edit Products')) SELECT * FROM HODTEST.EX01 WHERE ((HODTEST.EX01.DESCRIPT='Edit Products'))

図 42. 同等の大文字および小文字の例

結果セクションの使用

結果セクションのフィールドは、SQL ステートメントで生成されるデータを SQLQuery アクションが使用する方法を制御します。データはグローバル変数に書 き込んだり、ファイルに書き込んだり、表示することができます。

グローバル変数 (\$HMLSQLUtil\$) へのデータの格納: SQLQuery アクションで生 成されるデータのデフォルト宛先は、グローバル変数 \$HMLSQLUtil\$ です。「結 果の出力先 (Output Result To)」リスト・ボックスで「ファイル」または「モニタ ー (Display)」を宛先に指定した場合でも、Host On-Demand ランタイムは、常に 正常な SQLQuery アクションの結果でこの変数を更新します。例えば、「ファイ ル」を宛先に指定すると、Host On-Demand は次のようにします。

- SQLQuery アクションで生成されるデータを指定したファイルに書き込む。さらに、
- データを \$HMLSQLUtil\$ に書き込む。

したがって、同じデータベース照会を 2 回行うことなく、データをファイルに書き 込む (または表示する) と同時に、データをマクロで使用することができます。

データを \$HMLSQLUtil\$ だけに 格納するには、「結果の出力先 (Output Result To)」リスト・ボックスを展開して、「\$HMLSQLUtil\$」をクリックします。

後続のマクロ・アクションで、\$HMLSQLUtil\$ に格納されたデータを使用するに は、\$HMLSQLUtil\$ に関連付けられた Host On-Demand マクロ・ユーティリティ ー・ライブラリー (HML ライブラリー) からメソッドを起動する必要があります。 183 ページの『\$HMLSQLUtil\$』を参照してください。

ファイルへのデータの書き込み: データをファイルに書き込むには、「結果の出力 先 (Output Result To)」リスト・ボックスを展開して「ファイル」をクリックしま す。

「ファイル名」リスト・ボックスに、ファイル・パスを入力します。 (ファイル・ パスはストリングであるため、適切なマクロ形式 (基本または拡張) でストリングを 入力してください)。または、「ブラウズ (Browse)」をクリックしてファイルを参照 します。注: このフィールドには、ドライブ名 (あれば) を含めた完全なファイル・ パスおよびファイル名を指定する必要があります (例えば、c:¥Documents and Settings¥User1¥output.txt)。

「ファイル・タイプ」リスト・ボックスで、データを格納するファイルのタイプを クリックします。このリスト・ボックスには、Database On-Demand クライアント で使用できるのと同じデータベース・ファイル・タイプがリストされます。

- ファイル・タイプが HTML の場合、Host On-Demand には、マクロ・エディ ターで直接アクセスできない追加の表示オプション (テーブルの枠、セル間隔、 列見出しなど) があります。ただし、SQL ウィザードを起動して、HTML オプ ションを変更し (SQL ウィザードの「出力 (Output)」タブの「設定 (Settings)」 をクリックする)、SQL ステートメントをマクロに保管することによって、これ らのオプションをマクロに組み込むことができます (121 ページの『SQL ウィ ザードの使用』を参照)。
- ファイル・タイプが XML の場合、Host On-Demand には、マクロ・エディタ ーで直接アクセスできない追加のオプション (文字エンコード) があります。た だし、SQL ウィザードを起動して、XML オプションを変更し (SQL ウィザード の「出力 (Output)」タブの「設定 (Settings)」をクリックする)、SQL ステート メントをマクロに保管することによって、このオプションをマクロに組み込むこ とができます (121 ページの『SQL ウィザードの使用』を参照)。

「上書き (Overwrite)」をクリックして既存ファイルの内容を上書きするか、「付加 (Append)」をクリックして既存ファイルにデータを付加します。ファイルが存在し ない場合は、上書きまたは付加操作で新規ファイルが作成されます。

「Web ブラウザーで表示 (Show in Web browser)」チェック・ボックスを選択する と、Host On-Demand はデータをファイルに書き込んでファイルを閉じてから、フ ァイルをデフォルトのブラウザーで表示します。ブラウザーには、ファイル・タイ プ用の (Microsoft Excel BIFF3 または BIFF4 など) プラグイン (例えば、Excel プ ラグイン) が必要です。一部のブラウザーでは、ブラウザーのセキュリティー設定 を構成し、Java アプレットまたはアプリケーションがブラウザーのインスタンスを 起動してローカル・ファイルを表示できるようにする必要があります (オンライ ン・ヘルプの『ローカル・ファイルを表示するブラウザーの構成 (Configuring a browser to display a local file)』を参照)。

データの表示: データを表示するには、「結果の出力先 (Output Result To)」リスト・ボックスを展開して「モニター (Display)」をクリックします。このオプションを選択すると、マクロ・ランタイムはデータを「照会結果 (Query Results)」ダイアログ内に表示します。

「出力ダイアログで待機 (Hold on output dialog)」チェック・ボックスは、「照 会結果 (Query Results)」ダイアログの表示後に、マクロ・ランタイムが次のマク ロ・アクションの処理に即時に進むかどうかを制御します。

- このチェック・ボックスを選択すると、マクロ・ランタイムは次のようになる。
 - 1. 「照会結果 (Query Results)」ダイアログに次の 3 つのボタンを表示する。
 - 結果の保管 (Save Results)
 - 継続 (Continue)
 - 閉じてから継続 (Close and Continue)
 - 2. エンド・ユーザーが 2 つ目 (「継続 (Continue)」) または 3 つ目 (「閉じ てから継続 (Close and Continue)」) のボタンをクリックするまで、次のマ クロ・アクションの処理に進まない。
- このチェック・ボックスをクリアすると、マクロ・ランタイムは次のようになる。
 - 1. 「照会結果 (Query Results)」ダイアログに次の 2 つのボタンを表示する。

- 結果の保管 (Save Results)
- 閉じる (Close)
- 2. 次のマクロ・アクションの処理に即時に進む。

どちらの場合も、「照会結果 (Query Results)」ダイアログは、エンド・ユーザーが 閉じる (「閉じる (Close)」または「閉じてから継続 (Close and Continue)」をク リックする) まで開いたままです。

表 19 は、各ボタンごとに名前、機能、およびボタンの表示に影響を与える「出力 ダイアログで待機 (Hold on output dialog)」チェック・ボックスの設定を示して います。

ボタン名: 機能: 「出力ダイアログで待機 (Hold on output dialog) チェック・ボックスの設定: 結果の保管 (Save Results) 選択またはクリア • データを ASCII 形式でフ ァイルに保管する 閉じる (Close) クリア ダイアログを閉じる。 継続 (Continue) • マクロ・ランタイムが次の | 選択 マクロ・アクションの処理 を開始する。 閉じてから継続 (Close & 選択 ダイアログを閉じる。 Continue) マクロ・ランタイムが次の マクロ・アクションの処理 を開始する。

表 19. 「照会結果 (Query Results)」ダイアログのボタン

Host On-Demand クライアント用のデータベース・コンポーネント のプリロード

デプロイメント・ウィザードで、データベースにアクセスするために Host On-Demand クライアントがプリロードするマクロ・コンポーネントを指定できま す。オンライン・ヘルプの『プリロード・オプション (Features1) (Preload Options (Features1))』を参照してください。

トレース・アクション (<trace> エレメント)

トレース・アクションは、Java コンソールなど、指定したトレース宛先にトレース・メッセージを送ります。

「トレース・ハンドラー (Trace Handler)」リスト・ボックスを使用して、トレー ス・メッセージを送信するトレース宛先を指定します。

- トレース・メッセージを Host On-Demand トレース機能に送るには、「Host On-Demand トレース機能 (Host On-Demand trace facility)」を選択します。
- トレース・メッセージをユーザー・トレース・ハンドラーに送るには、「ユーザ ー・トレース・イベント (User trace event)」を選択します。
- トレース・メッセージを Java コンソールに送るには、「コマンド行 (Command line)」を選択します。

トレース宛先に送るストリングを指定するには、「トレース・テキスト (Trace Text)」入力フィールドを使用します。

例

次の例では、トレース・メッセージを Java コンソールに送る方法を示します。こ の例では、トレースと変数更新のアクション・エレメントを使用します。

このマクロ・スクリプトのテキストを本書からシステム・クリップボードにコピー し、システム・クリップボードからコード・エディターにコピーできます (204 ペ ージの『本書からコード・エディターへのスクリプトのコピー・アンド・ペース ト』を参照)。このスクリプトをマクロ・エディターに保管した後、マクロ・エディ ターまたはコード・エディターを使用してスクリプトを編集できます。

この例については、次の点に注意してください。

- この例は、TRACE という名前の 1 つのマクロ・スクリプト全体で構成されます。
- <create> エレメントが \$strData\$ という名前のストリング変数を作成し、オリジナル値 'Original value' に初期化します。
- 最初のアクションは、トレース・テキストを 'The value is' + \$strData\$ に設 定したトレース・アクションです。
- 2 番目のアクションは、変数 \$strData\$ を新しい値 'Updated value' に設定する 変数更新アクションです。
- 3番目のアクションは、最初のトレース・アクションと同じもう1つのトレース・アクションです。

```
<HAScript name="TRACE" description=" " timeout="60000" pausetime="300"</pre>
            promptall="true" author="" creationdate="" supressclearevents="false"
             usevars="true" ignorepauseforenhancedtn="false"
             delayifnotenhancedtn="0">
   <vars>
      <create name="$strData$" type="string" value="'Original value'" />
   </vars>
   <screen name="Screen1" entryscreen="true" exitscreen="false" transient="false">
      <description>
          <oia status="NOTINHIBITED" optional="false" invertmatch="false" />
     </description>
      <actions>
         <trace type="SYSOUT" value="'The value is '+$strData$" />
         <varupdate name="$strData$" value="'Updated value'" />
         <trace type="SYSOUT" value="'The value is '+$strData$" />
      </actions>
      <nextscreens timeout="0" >
     </nextscreens>
 </screen>
```

図 43. サンプル・コード TRACE

このスクリプトにより、マクロ・ランタイムは次のデータを Java コンソールに送 ります。 The value is +{\$strData\$ = Original value} The value is +{\$strData\$ = Updated value}
上記のトレース出力の中で、\$strData\$ の値を単に表示する代わりに、デバッグ・ア クションが変数の名前とその値の両方を中括弧 {} の中に表示している点に注意して ください。

ユーザー・トレース・イベント

ユーザー・トレース・イベントの設定を利用するには、Host Access Toolkit 製品が 別途必要です。 MacroRuntimeListener インターフェースをインプリメントする必 要があります。マクロ・ランタイムは、次のタイプの発生に関するイベントを MacroRuntimeListeners に送ります。

- マクロ・エラー
- マクロの状態変更
- トレース・アクション (イベントは MacroTraceEvent)
- プロンプト・アクション
- メッセージ・アクション
- 抽出アクション

変数更新アクション (<varupdate> エレメント)

<varupdate> エレメントは、変数に値を格納します。次の項目を指定する必要があります。

- 変数の名前
- ・ 変数に格納する値

マクロの再生中に、マクロ・ランタイムは変数更新アクションを実行して、指定の 値を指定の変数に格納します。

<description> エレメント内で変数更新アクションを使用することもできます (76 ページの『記述内の変数更新アクションの処理』を参照)。

値には演算式を使用でき、また変数とインポートしたメソッドの呼び出しを値に含 めることができます。値が式の場合、マクロの再生中にマクロ・ランタイムは式を 評価し、結果の値を指定の変数に格納します。

変数更新アクションは、プログラミング言語の代入ステートメントと同様に機能し ます。 Java プログラムでは、次のような代入ステートメントを書くことができま す。

boolVisitedThisScreen = true; intVisitCount = intVisitCount + 1; dblLength = 32.4; strAddress ="123 Hampton Court";

変数更新アクションでは、「変数更新 (Variable Update)」ウィンドウの「名前 (Name)」フィールドに式の左側 (変数) を入力し、同じウィンドウの「値 (Value)」 フィールドに式の右側 (値) を入力します。上記の Java 代入ステートメントに相当 するものを作成するには、次のように入力します。

表 20. 変数名と値の例

「名前 (Name)」入力フィールド:	「値 (Value)」入力フィールド:
\$boolVisitedThisScreen\$	true

表 20. 変数名と値の例 (続き)

「名前 (Name)」入力フィールド:	「値 (Value)」入力フィールド:
\$intVisitCount\$	\$intVisitCount\$+1
\$dblLength\$	32.4
\$strAddress\$	'123 Hampton Court'

指定する値は、コンテキストに応じた正しいデータ型、またはその型に変換可能な データ型 (40 ページの『自動データ型変換』を参照)に属している必要がありま す。

変数更新アクションは、次の点で非常に便利です。

- 「値 (Value)」フィールド内のエンティティーに式を使用できる。
- 式はアクションが実行されるまで評価されない。

式について詳しくは、 33 ページの『第 5 章 データ型、演算子、および式』を参 照してください。

フィールド変数に対する変数更新アクション

変数更新アクションを使用したフィールド変数の更新は、セッション・ウィンドウ 内の 3270 または 5250 フィールドの内容を読み取り、フィールドの内容をストリ ングとして変数に格納するために便利な方法です。

フィールド変数は、ストリング変数の一種です。フィールド変数は、ストリング変 数と同様にストリングを格納し、ストリング変数が有効であるすべてのコンテキス トでフィールド変数を使用できます。ただし、フィールド変数にストリングが格納 される方法について、フィールド変数とストリング変数は異なっています。フィー ルド変数に格納されるストリングは常に、マクロ・ランタイムが現行セッション・ ウィンドウ内の 3270 または 5250 フィールドから読み取ったストリングです。

変数更新アクションを使用してストリング変数を更新する際には、「変数更新 (Variable update)」ウィンドウに次の情報を指定します。

- フィールド変数の名前 (例: \$fldTmp\$)
- 位置ストリング (例: '5,11') (位置ストリングは、セッション・ウィンドウの行と 列の位置を示す、コンマで区切られた 2 つの整数からなるストリングです)

マクロ・ランタイムが変数更新アクションを実行すると、マクロ・ランタイムは次 の処理を行います。

- 1. 変数がフィールド変数であることを認識する。
- 2. フィールド変数の更新に使用する位置ストリングを検索する。
- 現行セッション・ウィンドウ内で、位置ストリングに指定された行と列の位置を 検索する。
- 4. 現行セッション・ウィンドウ内で、その行と列の位置にある 3270 または 5250 フィールドを検索する。
- 5. 先行ブランクと末尾ブランクを含む、3270 または 5250 フィールドの内容全体 を読み取る。
- 6. フィールドの内容全体をストリングとしてフィールド変数に格納する。

その後、ストリングが有効であるすべてのコンテキストでフィールド変数を使用で きます。例えば、次のようにフィールド変数を別のストリングと連結できます。 'The field¥'s contents are'+ \$fldPrint0ption\$

例として、次の特性を持つ 3270 または 5250 フィールドがセッション・ウィンド

- ウに含まれているとします。行 5、列 8 から始まる。
- 行 5、列 32 で終わる。
- ストリング 'Print VTOC information' を含む。

次の値を指定して変数更新アクションをセットアップします。

- 「変数更新 (Variable update)」ウィンドウの「名前 (Name)」フィールドに、作成したばかりのフィールド変数の名前 \$f1dData\$ を入力する。
- 「値 (Value)」フィールドに位置ストリング '5,11' を入力する。指定する必要 がある行と列の位置は 1 つだけであることと、その位置はフィールド内にある どの行と列の位置でもよいことに注意してください。

マクロ・ランタイムがこの変数更新アクションを実行すると、マクロ・ランタイム はフィールドの内容全体を読み取り、内容をストリングとして \$fldData\$ に格納し ます。これで、フィールド変数 \$fldData\$ にはストリング 'Print VTOC information' が格納されます。

フィールドの一部の読み取り: 変数更新アクションにフィールド変数を使用する場合に、2 つの位置を含む位置ストリングを指定できます。この機能は、フィールドの内容の一部のみを読み取りたい場合に使用します。

「値 (Value)」フィールドに、最初の位置と 2 番目の位置をコロン (:) で区切って 入力します。例えば、最初の位置が 5,14 で 2 番目の位置が 5,17 ならば、 '5,14:5,17' と入力します。

2 つの位置を指定する場合には、次のようになります。

- 最初の位置は、フィールド内で読み取る最初の位置を指定します。
- 2 番目の位置は、フィールド内で読み取る最後の位置を指定します。

例として、次の特性を持つ 3270 または 5250 フィールドがセッション・ウィンド ウに含まれているとします。

- 行 5、列 8 から始まる。
- 行 5、列 32 で終わる。
- ストリング 'Print VTOC information' を含む。

また、次の値を指定して変数更新アクションをセットアップしたとします。

- 「変数更新 (Variable update)」ウィンドウの「名前 (Name)」フィールドに、作成したばかりのフィールド変数の名前 \$f1dData\$ を入力する。
- 「値 (Value)」フィールドに位置ストリング '5,14:5,17' を入力する。ここで は、フィールド内の開始位置と終了位置の両方を指定しています。

マクロ・ランタイムがこの変数更新アクションを実行すると、マクロ・ランタイム はストリング 'VTOC' をフィールドから読み取り(最初の位置ストリングによって 指定した位置から始まり、2 番目の位置ストリングによって指定した位置まで続く)、ストリング 'VTOC' を \$fldData\$ に格納します。

2 番目の位置がフィールドの最後を越えている場合、マクロ・ランタイムは最初の 位置からフィールドの最後までストリングを読み取ります。その後マクロ・ランタ イムは、このストリングをフィールド変数に格納します。

Xfer アクション (<filexfer> エレメント)

Xfer アクション (「転送アクション」または「ファイル転送アクション」と読む) は、ワークステーションからホストに、またはホストからワークステーションにフ ァイルを転送します。

基本パラメーター

「転送方向 (Tranfer Direction)」リスト・ボックスには、ファイルをワークステー ションからホストに転送するか (「送信 (Send)」)、ホストからワークステーション に転送するか (「受信 (Receive)」) を指定する必要があります。「式 (Expression)」を選択した場合は、実行時に Send または Receive に解決される式 を指定する必要があります (例えば、変数 \$strDirection\$)。

表 21 に、「ホスト・ファイル名 (Host-File Name)」フィールドと「ローカル・フ ァイル名 (Local-File Name)」フィールドで使用する必要がある値を示します。

表 21.	「ホスト・	ファイル名	(Host-File N	Jame)」フィ	ィールドと	「ローカル・	ファイル名
(Local-Fil	e Name)]	フィールド					

転送方向:	「ホスト・ファイル名	「ローカル・ファイル名
	(Host-File Name)」	(Local-File Name)」
	フィールド:	フィールド:
送信 (Send)	ファイルがホストに着信した	ホストに送信するファイルの
	ときにファイルに割り当てる	名前。例えば、
	名前。例えば、3270 ディス	'e:¥¥tm¥¥trace1.txt'
	プレイ・セッションの場合	
	は、 'trace1 txt a'	
受信 (Receive)	ワークステーション側で受け	ファイルがクライアントに着
	取るファイルの名前。例え	信した後でファイルに割り当
	ば、'january archive a'	てる名前。例えば、
		'd:¥¥MyData¥¥january.arc'

拡張マクロ形式を使用する場合、円記号 ¥ は特殊記号なので '¥¥' と入力する必要 があります (34 ページの『拡張マクロ形式のストリング表記規則』を参照)。

高度なパラメーター

通常、「転送前にクリア (Clear Before Transfer)」フィールドでは、3270 ディスプ レイ・セッションの場合は true、5250 ディスプレイ・セッションの場合は false を使用する必要があります。

「タイムアウト (Timeout)」フィールドには、マクロ・ランタイムが転送を終了す る前に待機するミリ秒数を設定する必要があります。デフォルトは 10000 ミリ秒 (10 秒) です。この「タイムアウト (Timeout)」フィールドにより、ファイルの転送 に使用しているセッションが突然切断されたためにマクロ・ランタイムがハングす る状況を防止できます。ファイルが非常に大きい場合や接続が低速の場合は、値を 大きくする必要が生じることがあります。

ご使用のホストが必要とする追加パラメーターがある場合は、「オプション (Options)」フィールドを使用する必要があります。これらのパラメーターは、ホス ト・システムのタイプごとに異なります。ホストが iSeries の場合、次のパラメー ターを「オプション (Options)」フィールドに追加する必要があります。

USERID(myuserid) PASSWORD(mypassword)

ワークステーションの文字セットからホストの文字セットへの変換、およびその逆 の変換を行うためにマクロ・ランタイムが使用するコード・ページ (マッピング・ テーブル)を選択するには、「ローカル・コード・ページ (Local Code-page)」フィ ールドを使用する必要があります。セッション構成に指定されているものと同じコ ード・ページ番号 (例: 437)を選択する必要があります。

BIDI セッション (アラビア語またはヘブライ語) 用のパラメーター コード・エディターを使用して設定できる、BIDI セッション (アラビア語またはヘ ブライ語) 用の追加パラメーターがあります (233 ページの『属性』)。

セッションの指定

「ホスト ID (Host ID)」フィールドを使用して、このアクションを実行するセッションを指定します。

- マクロを起動するセッションでこのアクションを実行する場合は、「ホスト ID (Host ID)」フィールドをブランクのままにする。
- 別のセッションでこのアクションを実行する場合は、「ホスト ID (Host ID)」 フィールドにセッション参照を入力する(196 ページの『ホスト ID の指定』を 参照)。

参照されたセッションがアクティブである場合は、このセッションに任意の自動編 集機能を使用できます (197 ページの『異なるセッションでの自動編集機能の使 用』を参照)。

例

次に、Xfer アクションの例を示します。

```
<actions>
<filexfer direction="send" pcfile="'c:¥¥myfile.txt'" hostfile="'myfile text A0'"
clear="true" timeout="10000" pccodepage="437" />
</actions>
```

図 44. Xfer アクションの例

第9章 画面認識、パート2

有効な次画面

43 ページの『第6章マクロ・ランタイムによるマクロ画面の処理方法』で説明 したとおり、マクロ・ランタイムは通常、現行マクロ画面の <nextscreens> エレメ ント内を検索して、次のマクロ画面になる候補であるマクロ画面の名前を見付けま す。つまり、マクロ画面自体の中に、次の処理対象として有効なマクロ画面のリス トが含まれています (入り口画面と一時画面は例外です。 139 ページの『入り口画 面、出口画面、および一時画面』を参照してください)。

マクロ・エディターでは、「リンク (Links)」タブをユーザー・インターフェースと して使用して、マクロ画面の <nextscreens> エレメントに候補マクロ画面の名前を 保管します。図 45 に、「リンク (Links)」タブの例を示します。

ホスト・アクセス・マクロ・エディター	×	
マクロ 画面 リング 変数		
画面名 Screen1 Timout 0 ミリ秒		
使用可能な画面 有効な次画面 Scroop2		
保管して終了 別名保管… 保管 キャンセル コード・エディター… インポート… エクスポート…	、ルプ	
定義済みの画面に対して有効な次画面を定義する		

図 45. 「リンク (Links)」 タブの例

前記の図で、タブの最上部にある「画面名 (Screen Name)」リスト・ボックスに は、マクロ全体のマクロ画面すべてのリストがあります。現在選択されているマク ロ画面は Screen1 です。右側の「有効な次画面 (Valid Next Screens)」リスト・ボ ックスは、Screen1 の候補マクロ画面をリストしています (Screen1 の <nextscreens> エレメントの名前があるこのリスト・ボックスと、マクロの再生時 にマクロ・ランタイムが使用する有効な次画面のリストを混同しないでください)。 左側の「使用可能な画面 (Available Screens)」リスト・ボックスは、他のマクロ画 面すべての名前をリストしています。 前の図では「使用可能な画面 (Available Screens)」リストに画面が 1 つしか示され ていませんが、これはこの図がマクロ画面を 2 つしか含まない (Screen1 と Screen2) マクロのものだからです。その代わりに、20 個の画面からなるマクロを考 え、新規マクロ画面 ScreenR の <nextscreens> リストにマクロ画面を追加したい とします。次の手順で行います。

- 1. 「リンク (Links)」タブで、「画面名 (Screen Name)」リスト・ボックスを展開 し、ScreenR が見付かるまでスクロールダウンする。
- 2. ScreenR を選択する。
- 3. ScreenR は新規画面なので、右側の「有効な次画面 (Valid Next Screens)」リ ストにマクロ画面名はリストされません。
- 左側の「使用可能な画面 (Available Screens)」リスト・ボックスには、マクロ 内のマクロ画面すべての名前があります。
- 5. ScreenR のリストに追加する画面を選択する。 ScreenS を選択したとします。
- ScreenS を選択した後、2 つのリスト・ボックスの間にある右矢印のボタンをク リックする。ScreenS が右側のリスト・ボックスに追加され、左側のリスト・ボ ックスから除去されます。
- 同じように、追加したい他のマクロ画面の名前を ScreenR の「有効な次画面 (Valid Next Screens)」リスト・ボックスに移動する。
- 8. ScreenS、ScreenG、および ScreenY の合計 3 つの画面名を移動したとしま す。

完了すると、現在選択されているマクロ画面 ScreenR の有効な次画面のリストに、 3 つのマクロ画面の名前が表示されます。

コード・エディターでは、有効な次のマクロ画面の名前

```
ScreenS、ScreenG、ScreenY が、図 46 に示すように ScreenR の中に保管されてい
ることが分かります。
```

図 46. <nextscreens> エレメントがあるマクロ画面 ScreenR

前の図では、ScreenR の <screen> エレメントの name 属性は "ScreenR" に設定 されています。内側には、<screen> エレメントの 3 つの基本構造エレメント (<description> エレメント、 <actions> エレメント、および <nextscreens> エレメ ント) があります。 <description> エレメントと <actions> エレメントの内容は示 されていませんが、省略符号 (...) として表されています。<nextscreens> エレメン トには 3 つの <nextscreen> エレメントがあり、それぞれの <nextscreen> エレメ ントには、有効な次画面 ScreenS、ScreenG、および ScreenY のいずれかの名前が あります。

ランタイム処理について詳しくは、 43 ページの『第6章 マクロ・ランタイムに よるマクロ画面の処理方法』を参照してください。

入り口画面、出口画面、および一時画面

入り口画面、出口画面、および一時画面の設定を使用して、特別な方法で扱うマク ロ画面をマクロ・ランタイムに指示できます。マクロ・エディターの「画面 (Screens)」タブの「一般 (General)」タブで、これらの設定を行います。タブの一番 上の「画面名 (Screen Name)」フィールドには、入り口画面、出口画面、および一 時画面のリスト・ボックスがあります。これらのリスト・ボックスそれぞれに対し て、ブール値 (デフォルトは false)、またはブール値として評価される式を指定する 必要があります。

コード・エディターでは、これらの設定値は <screen> エレメントの属性として表示されます。前記の 138 ページの図 46 では、entryscreen、exitscreen、および transient の 3 つの属性が ScreenR の <screen> エレメントにあることが分かります。

入り口画面

マクロの再生時に最初に処理される画面として扱うマクロ画面を指定するには、 「入り口画面 (Entry Screen)」を true に設定します。ただ 1 つのマクロ画面、または複数のマクロ画面を、入り口画面として指定できます。

マクロの再生が開始されると、マクロ・ランタイムはマクロ・スクリプトを検索 し、入り口画面として指定されたマクロ画面すべてを検出します。その後、マク ロ・ランタイムはこれらの入り口マクロ画面の名前を、有効な次画面のランタイ ム・リストに追加します。最後にマクロ・ランタイムは、リストにある画面のどれ かと現行セッション・ウィンドウの一致を通常の方法で検索します。

入り口マクロ画面のどれかとセッション・ウィンドウの一致をマクロ・ランタイム が検出すると、そのマクロ画面が最初に処理されるマクロ画面になります。最初の マクロ画面のアクションを実行する前に、マクロ・ランタイムは、有効な次画面の ランタイム・リストから入り口マクロ画面の名前を除去します。

複数の入り口画面があるマクロ

同じマクロに複数の入り口画面がある状況の 1 つは、ホスト・アプリケーションが アプリケーション画面 A を開始し、次にアプリケーション画面 B、その次にアプ リケーション画面 C といったように、一連のアプリケーション画面を次々に開始す るときです。例えば、画面 A はログオン画面、画面 B は複数のサポート・プロセ スを開始する画面、画面 C はアプリケーションの実際の初期画面などです。

この状況では、ユーザーがアプリケーション画面 A、B、または C のどれを表示していてもマクロを実行できるようにしたい場合があります。

入り口画面は通常画面にもすることが可能

画面を入り口画面として指定しても、その画面はマクロ内で通常の画面としても使用でき、他のマクロ画面の <nextscreens> リストに含めることができます。

例えば、一連のメニュー選択項目のある、中央アプリケーション画面をもつホス ト・アプリケーションがあるとします。メニューの選択を行うたびに、アプリケー ションはいくつかのアプリケーション画面の処理を進め、元の中央アプリケーショ ン画面に戻ります。

この状況で、マクロ ScreenA が中央アプリケーション画面に対応するマクロ画面で あるとします。この場合、次のようになります。

- マクロは中央アプリケーション画面から開始できるので、ScreenA を入り口画面 にすることができます。
- ScreenA は他のマクロ画面の <nextscreens> エレメント内でも指定できます。
 これは、それぞれのタスク後にアプリケーションが中央アプリケーション画面に
 戻るからです。

出口画面

マクロ画面の「出口画面 (Exit Screen)」を true に設定すると、マクロ・ランタイ ムはそのマクロ画面のアクションを実行した後、マクロを終了します。つまり、マ クロ・ランタイムがアクションを実行した後、画面認識に進む前に、マクロ・ラン タイムは現行マクロ画面の出口画面標識が true に設定されているかどうか調べま す。 true に設定されている場合、マクロ・ランタイムはマクロを終了します (マク ロ・ランタイムは、出口画面の <nextscreens> エレメントを無視します)。

このため、マクロの終了ポイントにしたいマクロ画面がある場合は、そのマクロ画面の「出口画面 (Exit Screen)」を true に設定します。

マクロの出口画面はいくつでも設定できます。複数の出口画面がある状況の例をい くつか示します。

- マクロに 1 つの正常終了ポイントと複数の異常終了ポイント (エラーが発生した 場合に進む) がある。
- 処理中に特定のポイントでマクロを停止することも、先に進むこともできるので、複数の正常終了ポイントが存在する。

一時画面

一時マクロ画面は、次の特性を持つアプリケーション画面の処理に使用されます。

- アプリケーションのフロー中、そのアプリケーション画面の発生が予測できない。複数のポイントで発生する場合もあれば、まったく発生しない場合もあります。
- そのアプリケーション画面に対して行う必要があるアクションは消去のみ。

このようなアプリケーション画面の一例は、ユーザーが無効なデータを入力したと きにアプリケーションが表示するエラー画面です。このエラー画面が表示される回 数は予測不能 (ユーザーが無効なデータを入力したとき) であり、マクロ開発者とし てこのエラー画面に対して行う必要があるアクションは、エラー画面を消去してマ クロを元どおり実行することだけです。 マクロ・ランタイムがマクロの再生を準備する際に、マクロ・ランタイムが入り口 画面の名前を有効な次画面のランタイム・リストに追加する時点で、マクロ・ラン タイムは一時画面として指定されたすべてのマクロ画面 (あれば) もリストの最後に 追加します。

これらの一時画面のリストは、マクロの再生全体を通して、有効な次画面のランタ イム・リストに残ります。マクロ・ランタイムが新しい候補マクロ画面の名前を (現行マクロ画面の <nextscreens> エレメントから) リストに追加するとき、マク ロ・ランタイムはこれらの新しい候補名を一時画面の名前の前に追加するので、一 時画面の名前は必ずリストの最後にあります。

マクロ・ランタイムが画面認識を実行するとき、リストにあるすべての名前のマク ロ画面を通常の方法で評価します。リストに名前がある候補マクロ画面から、アプ リケーション画面との一致をマクロ・ランタイムが検出できない場合、マクロ・ラ ンタイムはリストを順に進めて、アプリケーション画面のリストに指定された一時 マクロ画面のどれかとの一致を検索します。

現行アプリケーション画面と一時マクロ画面のどれかの一致を検出した場合、マク ロ・ランタイムはリストから名前を除去しません。代わりに、マクロ・ランタイム は一時マクロ画面のアクション (予期しないアプリケーション画面を消去する)を実 行し、予期しないアプリケーション画面が発生したときに行っていた画面認識処理 に戻ります。

一時画面の処理の例

マクロ・ランタイムが画面認識を行っていて、有効な次画面のリストに 3 つのマク ロ画面の名前があるとします。これらの画面は、ScreenB と ScreenD (候補画面の 名前)、および ScreenR (一時画面の名前) です。マクロ・ランタイムは次のステッ プを実行します。

- 1. セッション・ウィンドウの表示スペースが更新されると、マクロ・ランタイムは 有効な次画面のリストにある名前を通常の方法で評価します。
- 2. 予期しないアプリケーション画面が発生したため、ScreenB または ScreenD は 現行アプリケーション画面と一致せず、ScreenR は現行アプリケーション画面 と一致したとします。
- 3. 一時画面が認識されたので、マクロ・ランタイムは有効な次画面のリストから名 前を除去しません。
- 4. マクロ・ランタイムは、ScreenR を処理対象の現行マクロ画面にします。
- 5. マクロ・ランタイムは、ScreenR のアクションを実行します。これらのアクションは、予期しないアプリケーション画面を消去します。
- 6. マクロ・ランタイムは、ScreenR 内の <nextscreens> エレメント (あれば) を 無視します。
- マクロ・ランタイムは、前述のステップ1 で行っていた直前の画面認識タスク に戻ります。有効な次画面のリストは変更されていません。今度は、予想通りの アプリケーション画面が表示され、ScreenD がその画面に一致することをマク ロ・ランタイムが検出したとします。この場合、次のようになります。
 - a. マクロ・ランタイムは、ScreenR を次に処理するマクロ画面にします。
 - b. マクロ・ランタイムは、有効な次画面のリストから名前 ScreenB と ScreenD を除去します。名前 ScreenR はリストに残ります。

c. マクロ・ランタイムは、ScreenD のアクションの処理を開始します。

画面認識のタイムアウト設定

ここでは、現行アプリケーション画面と有効な次画面のリストにある画面の一致が 見付からないために、マクロ・ランタイムが先に進むことができないシナリオにつ いて説明します。次の 2 つのフィールドを使用して、タイムアウト値を設定できま す。タイムアウトになる前に画面認識が成功しない場合、マクロは終了します。

- 「マクロ (Macro)」タブの「画面間のタイムアウト (Timeout Between Screens)」フィールド
- 「リンク (Links)」タブの「タイムアウト (Timeout)」フィールド

画面認識

前述のとおり、マクロ・ランタイムがマクロ画面の <actions> エレメントのアクションをすべて実行した後、マクロ・ランタイムは、有効な次画面のリストにある画面のどれかと新規アプリケーション画面の一致を検索します (43 ページの『第 6章 マクロ・ランタイムによるマクロ画面の処理方法』を参照)。

場合によっては、予測できない事情のために、有効な次画面のリストにあるマクロ 画面とアプリケーション画面の一致を、マクロ・ランタイムが検出できないことが あります。例えば、マクロ開発者が予測しなかったアプリケーション画面に進む入 カシーケンスをユーザーが入力する場合があります。また、システム・プログラマ ーがアプリケーション画面を変更したために、そのアプリケーション画面が、対応 するマクロ画面の <description> エレメント内の記述と一致しなくなることもあり ます。

こうしたシナリオが発生すると、マクロ・ランタイムが一致の検出を試行して失敗 し続けている間、マクロはハング状態になります。

画面間のタイムアウト (「マクロ (Macro)」タブ)

「画面間のタイムアウト (Timeout Between Screens)」チェック・ボックスと入力 フィールドは「マクロ (Macro)」タブにあり、画面認識のタイムアウト値を指定し ます。デフォルトでは、チェック・ボックスを使用可能にすると、この値はマクロ 内のそれぞれのマクロ画面に適用されます。ただし、「リンク (Links)」タブの「タ イムアウト (Timeout)」フィールドを使用して、特定のマクロ画面に対する値を変 更できます (次のセクションを参照)。

マクロ・ランタイムは、画面認識を開始するときに、マクロ全体に対して「画面間 のタイムアウト (Timeout Between Screens)」の値が設定されているかどうか、お よびマクロ画面に対する「タイムアウト (Timeout)」の値が設定されているかどう かを検査します。タイムアウト値が設定されている場合、マクロ・ランタイムは、 タイムアウト値に指定されているミリ秒数にタイマーを設定します。マクロ・ラン タイムが画面認識を完了する前にタイマーが満了した場合、マクロ・ランタイムは マクロを終了し、次のようなメッセージを表示します。 Macro timed out: (Macro=ispf_ex2, Screen=screen_address_type)

図 47. 画面認識タイムアウトのエラー・メッセージ

このメッセージには、マクロの名前と、タイムアウトが発生したときに処理してい た画面の名前が表示されていることに注意してください。例えば、このメッセージ に指定されている画面が ScreenA ならば、マクロ・ランタイムはすでに ScreenA のアクションをすべて実行済みで、ScreenA の有効な次画面リストにあるマクロ画 面とアプリケーション画面の一致を検索していました。

「画面間のタイムアウト (Timeout Between Screens)」フィールドを使用するに は、チェック・ボックスを選択し、マクロを終了する前に待つミリ秒数の値を入力 します。デフォルトでは、このチェック・ボックスにはチェックマークが付いてお り、タイムアウト値は 60000 ミリ秒 (60 秒) に設定されています。

タイムアウト (「リンク (Links)」タブ)

「リンク (Links)」タブの「タイムアウト (Timeout)」入力フィールドは、特定のマ クロ画面に対する画面認識のタイムアウト値を指定します。この値が 0 以外の場 合、マクロ・ランタイムは、「マクロ (Macro)」タブの「画面間のタイムアウト (Timeout Between Screens)」フィールドに設定されている値を使用せずに、この値 をこのマクロ画面の画面認識のタイムアウト値 (ミリ秒) として使用します。

マクロ・ランタイムが画面認識を完了する前にタイマーが満了した場合、マクロ・ ランタイムは図 47 のメッセージを表示します。

認識限度 (「画面 (Screens)」タブの「一般 (General)」タブ)

認識限度は、<screen> エレメントの開始タグ内の属性ではなく、<screen> エレメ ント内でオプションとして指定できる別個のエレメント (<recolimit> エレメント) です (<description>、<actions>、および <nextscreens> の各エレメントと同レベ ル)。

「認識限度の設定 (Set Recognition Limit)」チェック・ボックスと「エラー前の画 面 (Screens Before Error)」入力フィールドは、「画面 (Screens)」タブの「一般 (General)」タブにあります (27 ページの図 15 を参照)。デフォルトでは、「認識 限度の設定 (Set Recognition limit)」チェック・ボックスはクリアされていて、入 力フィールドは使用不可です。このチェック・ボックスを選択すると、マクロ・エ ディターは「エラー前の画面 (Screens Before Error)」入力フィールドのデフォルト 値を 100 に設定します。この値の設定は増減できます。

認識限度を設定すると、マクロ・ランタイムが特定のマクロ画面を処理する回数が 多すぎる場合に、何らかのアクションを行うことができます。マクロ・ランタイム が同じマクロ画面を処理する回数が多い場合 (100 など)、その理由はおそらく、マ クロ内でエラーが発生して、マクロがエンドレス・ループから抜け出せなくなって いるためです。 認識限度に達すると、マクロ・ランタイムはエラー・メッセージを出してマクロを 終了するか (これがデフォルト・アクションです)、指定した別のマクロ画面の処理 を開始します。

認識限度は1 つの特定の画面に適用され、デフォルトでは認識限度は設定されない ことに注意してください。認識限度は任意のマクロ画面に対して指定でき、認識限 度を組み込むそれぞれのマクロ画面ごとに、指定する認識限度の値は同じでも異な っていてもかまいません。

認識限度に達したことの判別

マクロ・ランタイムは、<recolimit> エレメントを含むそれぞれのマクロ画面ごとに 認識カウントを記録します。マクロの再生が開始されると、すべてのマクロ画面の 認識カウントが 0 になります。

マクロに ScreenB という名前のマクロ画面が含まれていて、ScreenB に認識限度 100 を指定した <recolimit> エレメントがあるとします。マクロ・ランタイムが ScreenB を認識するたびに (つまり、マクロ・ランタイムが ScreenB を次に処理す るマクロ画面として選択するたびに)、マクロ・ランタイムは次のステップを実行し ます。

- 1. マクロ・ランタイムは、ScreenB 内の <recolimit> エレメントの存在を検出す る。
- 2. マクロ・ランタイムは、ScreenB の認識カウントを増分する。
- 3. マクロ・ランタイムは、認識カウントを認識限度と比較する。
- 4. 認識カウントが認識限度より小さければ、マクロ・ランタイムは通常どおり ScreenB のアクション・エレメントの実行を開始する。
- ただし、認識カウントが認識限度より大か等しい場合、マクロ・ランタイムは <recolimit> エレメントに指定されたアクションを実行する。この場合、マク ロ・ランタイムは ScreenB のアクション・エレメントを処理しません。

認識限度に達したときのアクション

認識限度に達したときのデフォルト・アクションとして、マクロ・ランタイムは次 のようなエラー・メッセージを表示し、マクロを終了します。

Recolimit reached, but goto screen not provided, macro terminating.

認識限度のアクションとして、マクロ・ランタイムが別のマクロ画面に進むように するには、コード・エディターを使用して goto 属性を <recolimit> エレメントに 追加し、ターゲット・マクロ画面の名前を属性の値として指定する必要があります (250 ページの『<recolimit> エレメント』を参照)。

goto 属性を使用すると、マクロ・ランタイムはマクロを終了せず、代わりに属性に 指定されたマクロ画面の処理を開始します。

ターゲット・マクロ画面はどの目的にも使用できます。考えられる用途には次のも のがあります。

- デバッグ用。
- マクロを終了する前にユーザーに対して情報メッセージを表示する。
- マクロの処理を継続する。

第10章 アクション、パート2:タイミングの問題

この章では、アクションの処理に関係するタイミングの問題についていくつか説明 し、これらの問題に対処するために使用できるリソースについて説明します。

アクション後の休止

ここでは、前のアクションに未完了の副次作用があるために、アクションが期待ど おりに実行されないシナリオについて説明します。

次の2 つの設定値を使用して、実行時にアクション後の休止を追加できます。

- 「マクロ (Macro)」タブの「アクション間の休止 (Pause Between Actions)」
- 「画面 (Screens)」タブの「一般 (General)」タブの「休止時間の設定 (Set Pause Time)」

アクションの処理速度

マクロ・ランタイムは人間のユーザーよりはるかに迅速にアクションを実行するの で、マクロの再生中に予測しない問題が発生し、アクションが予想通りに実行され ない可能性があります。この原因は、前のアクションへの依存関係が生じることで す。

アプリケーション画面を切り替えるキー・ストロークを例に取ります。アプリケー ション画面がすでに切り替わっていることを後続のアクションが予期しているにも かかわらず、実際にはアプリケーション画面がまだ更新の途中である場合、後続の アクションは失敗します。

他の状況でも、マクロ・ランタイムがそれぞれのアクションを前のアクションの直 後に実行すると、アクション間のタイミングに依存するエラーが発生する場合があ ります。

アクション間の休止 (「マクロ (Macro)」タブ)

「マクロ (Macro)」タブの「アクション間の休止 (Pause Between Actions)」フィ ールドによって、次のような場合でのマクロ・ランタイムの待ち時間を指定しま す。

- 入力アクションまたはプロンプト・アクションの実行後に、これら 2 つのアクションのいずれかで起こりうる副次作用を完了させる。さらに、
- マクロ画面の最後のアクションの実行後、アクション処理のその他の副次効果を 完了させる。

当初、「アクション間の休止 (Pause Between Actions)」は、入力およびプロンプト・アクションの後だけでなく、すべて のタイプのアクションの後の休止として実装されていました。現在では、次のように実装されています。

• マクロ・ランタイムは、次の場合に待機します。

- マクロ画面の最後の入力またはプロンプト・アクション以外のすべての入力 またはプロンプト・アクションの後、インターバルが休止時間の 50% になっ た場合。
- マクロ画面の最後のアクションの後、インターバルが休止時間の 100% になった場合。
- マクロ・ランタイムは、次の場合には待機しません。
 - マクロ画面の最後の入力アクションまたはプロンプト・アクションの後 (その アクションがマクロ画面の最後のアクションではない場合)。
 - その他のすべてのタイプのアクションの後。

デフォルトでは、「アクション間の休止 (Pause Between Actions)」チェック・ボ ックスは使用可能になっており、タイムアウト値は 300 ミリ秒に設定されていま す。したがって、マクロ・ランタイムは、デフォルトで次の処理を行います。

- マクロ画面の最後の入力またはプロンプト・アクション以外のすべての入力また はプロンプト・アクションの後、150 ミリ秒待機する。
- マクロ画面の最後のアクションの後、300 ミリ秒待機する。

「アクション間の休止 (Pause Between Actions)」はすべてのマクロ画面に影響す ることに注意してください。つまり、この 1 項目を設定することで、問題がある可 能性のあるマクロ画面を個別に変更しなくても、マクロ全体のタイミング・エラー を回避できます。

休止時間の設定 (「画面 (Screens)」タブの「一般」タブ)

特定のマクロ画面の「アクション間の休止 (Pause Between Actions)」を延長また は短縮したい場合、または「アクション間の休止 (Pause Between Actions)」が重 要な意味を持つマクロ画面が少ししかない場合には、「画面 (Screens)」タブの「一 般」タブにある「休止時間の設定 (Set Pause Time)」設定値を使用できます。

デフォルトでは、このチェック・ボックスは使用不可になっています。

マクロ画面でこの設定値を使用可能にすると、マクロ・ランタイムはこの特定のマ クロ画面内で、指定されたミリ秒数を「アクション間の休止 (Pause Between Actions)」に使用します。

例えば、ScreenA の「休止時間の設定 (Set Pause Time)」チェック・ボックスを選 択し、値を 500 ミリ秒に設定した場合、マクロ・ランタイムは ScreenA の最後の 入力またはプロンプト・アクション以外の各入力またはプロンプト・アクションの 後に 250 ミリ秒間待ち、ScreenA 内の最後のアクションの後に 500 ミリ秒間待ち ます。

「休止時間の設定 (Set Pause Time)」を使用可能にしたマクロ画面をマクロ・ラン タイムが処理する際には、「マクロ (Macro)」タブの「アクション間の休止 (Pause Between Actions)」オプションの設定値は無視し、「休止時間の設定 (Set Pause Time)」の設定値のみを使用します。

特定のアクションの後に休止を追加する

マクロ画面内で特定のアクションの後に追加の休止が必要な場合は、そのアクションの後に休止アクションを追加できます。休止アクションに指定した待ちは、アク ション間の休止または休止時間の設定によって発生する待ちに追加されます。

画面の完了

次のマクロ画面の認識が早すぎる

ホストが新しいアプリケーション画面の表示を完全に終了する前に、マクロ・ラン タイムが ScreenB のアクションの処理を開始するというバグが、マクロ画面 ScreenB にあるとします。このタイミングの異常が問題になる状況はほとんどない かもしれませんが、例えば ScreenB の最初のアクションが抽出アクションであり、 マクロ・ランタイムがこのアクションによってアプリケーション画面の行 15 と 16 からデータを読み取るとします。ホストが新規データを行 15 から 16 にすべて書 き込む時間を待たず、マクロ・ランタイムがこのアクションを実行したとします。

この問題を分析すると、次のことが確認されます。

- セッションは、デフォルト接続 TN3270 を使用する 3270 ディスプレイ・セッションである。
- 次のアクションのシーケンスが実行される。
 - 1. 直前のマクロ画面の処理中に、マクロ・ランタイムは Enter キーをホストに 送る入力アクションを実行する。
 - ホストは Enter キーを受信し、新しいアプリケーション画面のためにコマン ドとデータの最初のブロックを送信する。
 - クライアントは最初のブロックを受信して処理し、これによりホスト・アプ リケーション画面の一部を更新するが、全部は更新しない。特に、アプリケ ーション画面の行 15 と 16 がまだ更新されていません。
 - 4. 一方でマクロ・ランタイムは、新しいアプリケーション画面に一致する有効 な次のマクロ画面の認識を開始する。
 - 5. コマンドとデータの最初のブロックによってアプリケーション画面が変更さ れた結果、マクロ・ランタイムはマクロ ScreenB を次に処理するマクロ画面 として認識する。
 - マクロ・ランタイムは ScreenB 内の最初のアクション・エレメントを実行する。これは、アプリケーション画面の行 15 と 16 からデータを読み取る抽出アクションです。
 - クライアントは 2 番目のコマンドとデータのブロックをホストから受信し、 処理する。これにより、行 15 と 16 を含む、アプリケーション画面の他の 部分が更新されます。

要約すれば、このタイミングの問題が起こった結果、ホストが行 15 と 16 の更新 を完了する前に、マクロ・ランタイムが新しいアプリケーション画面の行 15 と 16 を読み取りました。

通常の TN3270 プロトコル

この問題が発生した理由は、拡張されていない TN3270 プロトコルに、ホスト・ア プリケーションの画面が完了したことを、ホストがクライアントに通知する手段が 組み込まれていないことです (TN3270 は、文字指向の接続である Telnet を基礎と して、画面指向のプロトコルである 3270 データ・ストリームをインプリメントし たものです)。このため、ホストはいくつかのデータ・ブロックをクライアントに送 ることができなくても、「OK、アプリケーション画面は完了したので、ユーザーが データを入力できるようになりました」と通知します。しかし、それぞれのブロッ クが着信するとき、このアプリケーション画面の最後のブロックであるかどうかの 指示はありません。クライアントの視点から見ると、次のようなイベントが発生し ます。

- コマンドとデータのブロックが着信する。クライアントは、入力禁止標識を設定 し、ブロックを処理して、新しいデータをセッション・ウィンドウの指定部分に 表示する。その後、クライアントは入力禁止標識をクリアし、待機する。
- 2. 30 ミリ秒が経過する。
- 別のコマンドとデータのブロックが着信する。クライアントは、前のステップ1 と同じようにブロックを処理する。このブロックにより、画面の別の部分が更新 される。クライアントは待機する。
- 4. 50 ミリ秒が経過する。

ホストが新しいホスト・アプリケーション・データ画面を完全に表示するまで、こ のプロセスは継続します。クライアントは、ホスト・アプリケーションの画面が完 了したかどうか分からない状態で待機し続けます (詳しくは、 43 ページの『第 6 章 マクロ・ランタイムによるマクロ画面の処理方法』を参照してください)。

このプロセスが、人間のオペレーターにとって問題になることはありません。その 理由はさまざまですが、ここでは重要ではありません。

ただしこのプロセスは、マクロ・ランタイムにとっては画面認識中に問題になりま す。マクロ・ランタイムが、画面認識中に画面が更新されるたび、および OIA イ ベントが発生するたびに(48ページの『評価のやり直し』を参照)、アプリケーシ ョン画面と一致する有効な次のマクロ画面を検索することを思い出してください。 このため、マクロ・ランタイムは画面が完全に更新される前に一致を検出する可能 性があります。例えば、アプリケーション画面の行 3 に "ISPF Primary Option Menu" という文字があるときに認識が発生するように、ストリング・ディスクリプ ターが指示しているとします。ホストが行 3 を更新してこれらの文字を表示したと きに、マクロ・ランタイムは一致が発生したと判断しますが、ホストがアプリケー ション画面の残りの更新を完了したかどうかは考慮しません。

解決策

この問題の解決法は3つあります。

- 記述にディスクリプターを追加する。
- Enter キーを送る入力アクションの後に遅延を挿入する(147 ページの『次のマ クロ画面の認識が早すぎる』のステップ1 を参照)。
- TN3270E のコンテンション解消機能を使用する。

これらの解決策について、以下のサブセクションで説明します。

ディスクリプターの追加

この方法は場合によっては機能しますが、込み入っていて信頼性がありません。 ScreenB の記述部分に十分なディスクリプターを追加することにより、アプリケー ション画面の重要な部分が更新されるまで、マクロ・ランタイムが ScreenB を認識 しないようにします。

入力アクションの後に遅延を挿入する

セッションが通常の TN3270 セッションである場合、またはコンテンション解消機 能を使用しない TN3270E セッションである場合には、遅延の挿入が最適な解決策 です。つまり、入力アクション (この例では ScreenA) によってホストが新しいア プリケーション画面を送信した後、数百ミリ秒以上の休止を挿入します。この遅延 により、マクロ・ランタイムが次のマクロ画面 (ScreenB) のアクションの処理を開 始する前に、ホストがアプリケーション画面を更新するための十分な時間を取るこ とができます。

このシナリオで、入力アクションの後に休止を挿入する方法はいくつかあります。

- 「アクション間の休止 (Pause Between Actions)」の遅延を増やす。
- ScreenA の「休止時間の設定 (Set Pause Time)」を増やす。この方式をお勧め します。ScreenA に対してのみ休止時間が延長されるので、影響を受けるのは ScreenA のみです。
- ScreenA の入力アクションの直後に休止アクションを追加する。この方式もお勧めします。ちょうど必要な場所に休止が挿入されます。
- ScreenB の最初のアクションとして休止アクションを追加する。シナリオによっては、この方式を選択する場合もあります。ただし、ScreenA の後に複数のマクロ画面が表示される可能性があって (ScreenB、ScreenC、ScreenD など)、これらの後続マクロ画面それぞれに画面の完了の問題が発生する場合は、この方式を使用すると、これらの後続マクロ画面それぞれに対して、最初のアクションとして休止を挿入する必要があります。前の項目で示した方式を使用して、1 つのマクロ画面 ScreenA に休止アクションを挿入した方が簡単です。

通常の TN3270 セッションと、コンテンション解消機能を使用可能にした TN3270E セッションの両方でマクロを実行する必要がある場合は、XML マクロ言 語に備わっているいくつかの属性が役立ちます。 150 ページの『画面の完了に関係 する属性』を参照してください。

TN3270E のコンテンション解消機能の使用

TN3270E (拡張) は、セッションが接続する LU または LU プールをユーザーが指 定できるようにした TN3270 プロトコルの拡張形式です。また、サーバーに ASCII モードで接続するため (例えば、ファイアウォールにログオンするため) のネットワ ーク仮想端末装置 (NVT) プロトコルもサポートしています。

コンテンション解消モードは TN3270E のオプショナル機能で、TN3270E サーバー のすべてではなく一部がこの機能をサポートしています。この機能は、ホストがア プリケーション画面の更新を完了したかどうかクライアントが認識できないという 問題を解決します。クライアントが TN3270E セッションを実行していて、コンテ ンション解消をサポートするサーバーに接続している場合は、ホストがアプリケー ション画面の更新を完了するまで、マクロ・ランタイムは新しいマクロ画面を認識 しません。 Host On-Demand では、TN3270 でなく TN3270E を使用するように 3270 ディス プレイ・セッションを設定できます。このためには、3270 ディスプレイ・セッショ ン構成パネルの「接続構成 (Connection configuration)」ウィンドウで、該当するラ ジオ・ボタンをクリックします。

サーバーがコンテンション解消をサポートしている場合には、通常、Host On-Demand は TN3270E サーバーと自動的にコンテンション解消モードで通信し ます。ただし、 HTML パラメーターによって、コンテンション解消モードを使用 不可に設定することもできます (オンライン・ヘルプの『NegotiateCResolution』を 参照)。

画面の完了に関係する属性

Host On-Demand は、以下の両方の環境でマクロの単一バージョンをサポートする 際にマクロ開発者が経験する問題に対処するために、3 つのエレメント属性を備え ています。

- コンテンションを解決しない環境 (TN3270 サーバーに接続したクライアント、 またはコンテンション解消を使用しない TN3270E サーバーに接続したクライア ントによってマクロが実行される。したがって、ホストがアプリケーション画面 を更新する時間を取るために、マクロ画面に休止アクションが必要になる場合が ある)。
- コンテンションを解決する環境 (コンテンション解消を使用する TN3270E サーバーに接続したクライアントによってマクロが実行される。したがって、ホストがアプリケーション画面を更新する時間を取るために休止アクションを必要とするマクロ画面はない)。

これらの属性は、コード・エディターを使用して追加する必要があります。

ignorepauseforenhancedtn=true/false

<HAScript> エレメントの ignorepauseforenhancedtn パラメーターを true に設 定すると、コンテンションを解決する環境でセッションが実行されている場合、マ クロの再生中にマクロ・ランタイムは休止アクション (<pause> エレメント)をス キップします。コンテンションを解決しない環境で実行するためのマクロを開発し (休止アクションを挿入)、コンテンションを解決する環境でもそのマクロを不要な遅 延を入れずに (休止アクションを無視して)実行する必要が生じた場合には、この属 性を使用できます。

この属性を true に設定すると、マクロ・ランタイムはコンテンションを解決しな い環境では休止アクションを処理しますが (指定されたミリ秒数だけ待つ)、コンテ ンションを解決する環境では休止アクションを無視します。

ただし、この属性を true に設定すると、マクロ・ランタイムはマクロ内の休止ア クション (<pause> エレメント) をすべてスキップします (アプリケーション画面 を更新する時間を取るために挿入された休止だけでなく)。次のサブセクションで は、この 2 次的な問題について説明します。

ignorepauseoverride=true/false

<pause> エレメントの ignorepauseoverride パラメーターを、特定の <pause> エ レメント内で true に設定すると、<HAScript> エレメント内で **ignorepauseforenhancedtn** 属性が true に設定されていても、マクロ・ランタイム はその <pause> エレメントを処理します (指定されたミリ秒数だけ待つ)。

コンテンションを解決する環境で <HAScript> エレメントの

ignorepauseforenhancedtn 属性を true に設定した場合でも、<pause> エレメン トをスキップせずに常に実行したい場合は、<pause> エレメント内でこの属性を true に設定します。

delayifnotenhancedtn=(ミリ秒)

<HAScript> エレメントの delayifnotenhancedtn パラメーターをゼロ以外の値に 設定すると、OIA (オペレーター情報域) が変更されたという通知をマクロ・ランタ イムが受け取ったときに、マクロ・ランタイムは指定されたミリ秒数だけ自動的に 休止します。

コンテンションを解決する環境 (休止アクションを挿入する必要がない) でマクロを 開発し、コンテンションを解決しない環境 (アプリケーション画面を完了する時間 を取るために、マクロ画面に休止アクションが必要になる場合がある) でもそのマ クロを実行する必要が生じた場合には、この属性を使用できます。

この属性を true に設定した場合、コンテンションを解決しない環境でマクロを実 行すると、OIA が変更されたという通知を受け取るたびに、マクロ・ランタイムは 指定されたミリ秒数だけ休止を挿入します。例えば、200 ミリ秒の休止を指定する と、マクロ・ランタイムは OIA が変更されるたびに 200 ミリ秒間待ちます。

OIA に対する変更の通知があるたびにマクロ・ランタイムが休止することを、要約 的に累積する効果により、マクロ・ランタイムが新しいマクロ画面のアクションの 処理を開始する前に、アプリケーション画面が完了します。マクロ・ランタイム は、コンテンションを解決しない環境でセッションが実行されていることを検出し たときにのみ、これらの余分の休止を挿入します。

この属性の制限は、マクロ・ランタイムがこれらの余分の休止をすべての画面の処 理中に追加することです (画面の更新が問題になっている画面の処理中だけでな く)。ただし、余分に費やされる待ち時間は少しです。また、より重要な点として、 この属性を使用すればコンテンションを解決しない環境にマクロを迅速に適応させ ることができます。個々の画面をテストして、画面更新の問題があるそれぞれの画 面に休止アクションを挿入する必要はありません。

第11章 変数とインポートした Java クラス

変数とインポート型の概要

変数は、マクロにプログラミングで高度な動作を加えるために役立ちます。変数を 使用すれば、値の格納、結果の保管、カウント、テキスト・ストリングの保管、結 果の記憶など、プログラミングに必要なあらゆることを実行できます。

標準データ型 (string、integer、double、boolean、および field) のいずれかに属す る変数を作成できます。

また、Java クラスを表すインポート型に属する変数も作成できます。その後クラス のインスタンスを作成し、インスタンスに対してメソッドを呼び出すことができま す。この能力により、Java ランタイム環境 (JRE) ライブラリー、Host Access Toolkit 製品のライブラリー、ユーザー自身がインプリメントしたクラスやライブラ リー、他のソースからの Java クラスやライブラリーなど、Java クラス・ライブラ リーに提供されている豊富な機能が利用できるようになります。

拡張マクロ形式が必要

変数を使用するには、マクロに対して拡張マクロ形式を使用する必要があります (33 ページの『マクロ形式の選択』を参照)。このため、基本マクロ形式のマクロに 変数を追加するには、マクロを拡張マクロ形式に変換するかどうか決定する必要が あります。基本マクロ形式の古いマクロが多数のユーザーに使用されていて、問題 なく機能している場合は、マクロを現状のままにしておきたい場合があります。

ただし、記録済みマクロはすべて基本マクロ形式で記録されていることに注意して ください。つまり、マクロを最近記録し、そのマクロの開発を始めようとしている 場合は、単に拡張マクロ形式への切り替えをまだ済ませていないだけかもしれませ ん。

マクロ・エディターは、基本マクロ形式のままのマクロに変数を定義しようとする と、ポップアップ・ウィンドウに次のメッセージを表示して、これら両方の状況に 対応します。

You are attempting to use an advanced macro feature. If you choose to continue, your macro will automatically be converted to advanced macro format. Would you like to continue?

図 48. 注意喚起メッセージ

作成するマクロで変数を使用する予定にしている場合は「はい (Yes)」をクリック し、基本マクロ形式のマクロを変換したくない場合は「いいえ (No)」をクリックし ます。

変数の有効範囲

それぞれの変数のスコープは、変数を作成する対象のマクロに関してグローバルで す。つまり、マクロにある任意のマクロ画面から、マクロの変数すべてにアクセス できます。マクロ画面内のアクションまたはディスクリプターがその変数にアクセ スするには、単に変数名を使用するだけです。

例えば、\$intPartsComplete\$ という名前の変数を 0 に初期化したとします。マクロ の進行に伴って、次のように変数を使用できます。

- ScreenC がタスクの部分 1 を完了し、変数更新アクションを使用して \$intPartsComplete\$ を増分する。
- ScreenG がタスクの部分 2 を完了し、変数更新アクションを使用して \$intPartsComplete\$ を増分する。
- 3. ScreenM の条件アクションが、部分 1 と 2 のどちらが現時点で完了している かテストする。結果に応じて、マクロは ScreenR または ScreenS を次に処理す るマクロ画面として予期する。
- ScreenS がタスクの部分 3 を完了し、変数更新アクションを使用して \$intPartsComplete\$ を増分する。
- 5. ScreenZ がメッセージ・アクションを使用して \$intPartsComplete\$ の値を表示 する。

この例では、いくつかの異なるマクロ画面のアクションが、変数 \$intPartsComplete\$の読み取りまたは書き込みを実行できます。

「変数 (Variables)」タブの概要

変数は 1 つの画面ではなくマクロ全体に属するので、それに見合うように変数用の 上位タブが別個に用意されています。「変数 (Variables)」タブを使用して、以下の ことができます。

- 変数の作成
- 変数の除去
- 新規変数型としての Java クラスのインポート

標準データ型に属する変数を作成するには、マクロ・エディターの「変数 (Variables)」タブを使用します。 155 ページの図 49 に、「変数 (Variables)」タブ の例を示します。

ホスト・アクセス・マクロ・エディター	×
ホスト・アクセス・マクロ・エディター マクロ 画画 リンク 美教 変数 Variable1(\$strUserName\$) 除去 名前 \$strUserName\$ 型 \$tring インボート 初期値	
保管して終了 別名保管 保管 キャンセル コード・エディター インポート エクスポート ヘルプ マクロ内で使用する変数を定義する	

図 49. 「変数 (Variables)」 タブ

上の図では、マクロ・エディターの「変数 (Variables)」タブが選択されています。 現在選択されている変数の名前 \$strUserName\$ が、「変数 (Variables)」リスト・ ボックスに表示されています。その他 3 つのフィールド、「名前 (Name)」入力フ ィールド、「型 (Type)」リスト・ボックス、および「初期値 (Initial Value)」入力 フィールドには、マクロ・ランタイムがこの変数を作成するために必要とする情報 があります。

「変数 (Variables)」リスト・ボックスには、このマクロ用に作成された変数すべて の名前があります。編集または除去する変数をこのリスト・ボックスから選択で き、また変数を新規に作成するための **<新規変数>** 項目もあります。

現在選択されている変数の項目が、別のストリングの後に括弧で囲んで表示されて いることに注意してください。

Variable1(\$strUserName\$)

ストリング Variable1 は、作成した変数の数を示す設定値で、マクロ・スクリプト には保管されません。変数の実名は \$strUserName\$ であり、変数を使用するときに は、マクロ全体でこの名前を単独で使用する必要があります。

変数名 \$strUserName\$ がドル記号 (\$) で囲まれていることにお気付きかもしれません。これは要件であり、マクロ内で変数を使用するときには、必ず変数名をドル記号 (\$) で囲む必要があります。

「名前 (Name)」入力フィールドは、現在選択されている変数 \$strUserName\$の名前を表示します。既存の名前に上書き入力することによって、変数名を変更できます。通常、このフィールドは、新規に作成した変数に名前を割り当てるためだけに

使用してください。後でいつでもこのパネルに戻ってこの変数の名前を変更できま すが (例えば \$strUserFirstName\$ に)、マクロ内のどこかで、アクションやディス クリプターにこの変数の以前の名前をすでに使用している可能性があるので注意し てください。この「変数 (Variables)」タブで名前を変更した場合は、マクロ内でこ の変数を使用したすべての場所を再検討して、以前の変数名を新しい変数名に変更 する必要があります。

任意の変数名を選択できますが、選択できる文字にはいくつかの制限があります (160 ページの『変数名と型名』を参照)。本書で行っているように、データ型の省 略形で始まる名前を選択する必要はありません (ストリング変数 \$strUserName\$の str のように)。

「型 (Type)」リスト・ボックスには、変数に使用可能な型がリストされ、新規変数 に使用する型をここから選択できます。標準型は、string、integer、double、 boolean、および field です。また、java.util.Hashtable などの Java クラスをイ ンポート型としてインポートすると、「型 (Type)」リスト・ボックスはこのインポ ート型を取り込み、図 50 に示すように使用可能な型のリストに追加します。

string integer double boolean field java.util.Hashtable

図 50. インポート型を宣言した後の「型 (Type)」リスト・ボックスの内容

このリスト・ボックスは、新規に作成した変数に型を割り当てるためだけに使用し てください。後でこのパネルに戻ってこの変数の型を別の型に変更できますが、変 数名の場合と同様に、マクロの中で最初に選択した型を必要とするコンテキスト で、すでにこの変数を使用している可能性があるので注意してください。この場 合、該当するそれぞれの個所を調べて、変数を使用しているコンテキストが新しい 型に適しているかどうか確認する必要があります。

「初期値 (Initial Value)」入力フィールドには、変数の初期値を指定できます。マ クロ・エディターは、型に応じて次のデフォルト値を提供します。

変数の型:	デフォルト初期値:
string	ストリングなし
integer	0
double	0.0
boolean	false
field	(初期値なし)
(任意のインポート型)	ヌル

表 22. 変数のデフォルト初期値

新しい初期値を指定するには、単にデフォルト値に上書きして入力します。

「除去 (Remove)」ボタンは、現在選択されている変数を除去します。

「インポート (Import)」ボタンと「インポート (Import)」ポップアップ・ウィンド ウについては、 158 ページの『Java クラスのインポート型の作成』で説明しま す。

変数の新規作成

マクロ・エディターで変数を新規に作成するには、まず「変数 (Variable)」リスト・ボックスの最後にある「<新規変数> (<new variable>)」項目をクリックします。マクロ・エディターは変数を新規に作成し、いくつかの初期特性を割り当てます。これらの特性は、ユーザーがニーズに合うように修正する必要があります。初期値は次のとおりです。

- 1. 初期名 (\$a1\$ など)。
- 2. 初期型 (string)。
- 3. 初期値。型によって異なります (156 ページの表 22 を参照)。

次に、新規変数に必要な値を設定する必要があります。例えば、画面をカウントす るための整変数を作成し、その初期値を 1 にする必要がある場合は、次のように初 期値を設定します。

- 1. 「名前 (Name)」入力フィールドに、名前 \$intScreenCount\$ を入力します。
- 2. 「型 (Type)」リスト・ボックスから、integer データ型を選択します。
- 3. 「初期値 (Initial Value)」フィールドに 1 を入力します。

「変数 (Variables)」タブのほかに、いくつかの便利な位置で、マクロ・エディター から変数を新規に作成するためのポップアップ・ウィンドウにアクセスできます。 例えば、変数更新アクションの場合、「名前 (Name)」リスト・ボックスにはすで に作成済みの変数名がすべて表示されるだけでなく、「**<新規変数> (<New** Variable>)」項目も表示されます。この項目をクリックすると、変数を新規に作成 するためのポップアップ・ウィンドウが開きます。このポップアップ・ウィンドウ を使用して作成した変数は、「変数 (Variables)」タブで作成した変数と同等です。

コード・エディターでは、<create> エレメントを使用して変数を新規に作成しま す。マクロ・スクリプト内の変数すべてを含む収容エレメント <vars> があり、そ れぞれの変数ごとに <create> エレメントがあります。図 51 に、5 つの <create> エレメントを含む <vars> エレメントを示します。

```
<vars>
    <create name="$strAccountName$" type="string" value="" />
    <create name="$intAmount$" type="integer" value="0" />
    <create name="$dblDistance$" type="double" value="0.0" />
    <create name="$boolSignedUp$" type="boolean" value="false" />
    <create name="$fldFunction$" type="field" />
</vars>
```

図 51. サンプル <vars> エレメント

上の図の <vars> エレメントは、それぞれの標準データ型 (string、integer、double、 boolean、および field) から変数を 1 つ作成します。 それぞれの <create> エレメントの属性は、「変数 (Variables)」タブのフィールド と一致していることに注意してください。name 属性は変数名を指定し、type 属性 は型を指定し、value フィールドは初期値を指定します。 変数の作成 (<create> エレメント) は、すべて <vars> エレメントの中で行う必要 があります。<vars> エレメント自体は、<import> エレメントがあればその後 (次 のセクションを参照)、かつ最初のマクロ画面 (<screen> エレメント) の前に置く必 要があります。

Java クラスのインポート型の作成

Host On-Demand マクロは、Java クラスをインポートする方法として、インポート型を使用します。つまり、インポート型をまず作成し、特定の Java クラスに関 連付ける必要があります。この作業は、マクロごとに、Java クラス 1 つに対して 1 回のみ行います。インポート型を作成するには、次の手順で行います。

- 1. 「変数 (Variables)」タブの「インポート (Import)」ボタンをクリックする。 「インポート (Import)」ポップアップ・ウィンドウが表示されます。
- 「インポート型 (Imported Types)」リスト・ボックスで、項目「<新規のインポート型> (<new imported type>)」を選択する。
- 3. 型のクラス名を入力する (例: java.util.Hashtable)。パッケージ名 (あれば) を含む、完全修飾クラス名を入力する必要があります。
- 短縮名を入力する (例: Hashtable)。短縮名を指定しない場合、マクロ・エディ ターは完全修飾クラス名を短縮名として使用します。短縮名を指定すると、イン ポート型を参照する際に、短縮名または完全修飾クラス名のどちらも使用できま す。
- 5. 「OK」をクリックする。

このインポート型に属する変数を作成するには、変数を通常の方法で作成します が、インポート型を変数の型として選択します。インポート型の変数を作成するに は、次の手順で行います。

- 「変数 (Variables)」リスト・ボックスの最後にある「<新規変数> (<new variable>)」項目をクリックする。マクロ・エディターは、名前 (\$a1\$ など)、 型 (string)、初期値 (ブランク) など、デフォルトの初期値を通常どおり表示し ます。
- 2. 「名前 (Name)」入力フィールドに、任意の名前 (例: \$ht\$) を入力する。
- 「型 (Type)」リスト・ボックスで、インポート型を選択する。例えば、 Hashtable (型をインポートしたときに短縮名を指定した場合)、または java.util.Hashtable (完全修飾クラス名と同じ、デフォルトの短縮名を受け入 れた場合)を選択します。
- 「初期値 (Initial Value)」フィールドでは、フィールドをブランクのままにする (初期値は null になる)か、クラスのインスタンスを戻すメソッドを指定でき ます。例えば、\$new Hashtable()\$ (短縮名を使用)、または \$new java.util.Hashtable()\$ (完全修飾クラス名を使用)を指定します。

コンストラクターはドル記号 (\$) で囲む点に注意してください。変数の名前をドル 記号で囲む必要があるように、Java メソッドの呼び出しもドル記号で囲む必要があ ります (これは、ドル記号で囲むことにより、項目を評価する必要があることをマ クロ・ランタイムに示すためです)。

「インポート (Import)」ポップアップ・ウィンドウに戻ると、「インポート型 (Imported Types)」リスト・ボックスを使用して型を新規に作成でき、すでに作成 済みの型を編集または削除することもできます。型を新規に作成するには、リスト の最後にある「<新規のインポート型> (<new imported type>)」をクリックしま

す。型を編集するには、「インポート型 (Imported Types)」リスト・ボックスから タイプを選択し、「クラス (Class)」と「短縮名 (Short Name)」の入力フィールド で値を変更します。型を除去するには、型を選択して「除去 (Remove)」をクリック します。

短縮名を指定する際には、特定の制限のもとで任意の名前を使用できます (160 ペ ージの『変数名と型名』を参照)。

コード・エディターでは、<type> エレメントを使用してインポート型を作成しま す。マクロ・スクリプト内のインポート型すべてを含む収容エレメント <import> があり、それぞれのインポート型ごとに <type> エレメントがあります。図 52 は、インポート型を宣言する <import> エレメント、およびインポート型に属する 変数を作成して初期化する後続の <vars> エレメントを示しています。

```
<import>
<type class="java.util.Hashtable" name="Hashtable" />
</import>
<vars>
<create name=$ht$ type="Hashtable" value="$new Hashtable(40)$" />
</vars>
```

図 52. インポート型とその型の変数

上の図で、<import> エレメントには <type> エレメントが 1 つ含まれ、このエレ メントには class 属性 (完全修飾クラス名 java.util.Hashtable を指定) と name 属性 (短縮名 Hashtable を指定) があります。<vars> エレメントには <create> エ レメントが 1 つ含まれ、このエレメントは名前 (\$ht\$)、型 (Hashtable)、および初 期値 (ここでは null ではなく、クラスのインスタンスを戻すコンストラクター \$new Hashtable(40)\$ の呼び出し) を通常どおり指定します。

コード・エディターを使用している場合は、インポート型 (<type> エレメント)を すべて <import> エレメント内に入れる必要があります。<import> エレメント自 体は、<HAScript> エレメントの内部 (234 ページの『<HAScript> エレメント』 を参照)、かつ <vars> エレメントの前に置く必要があります。

注意を必要とする問題

Java ライブラリーまたはクラスの配置

マクロの再生中に、マクロ・ランタイムが Java メソッドの呼び出しを処理する と、マクロ・ランタイムは、クラスを検出するまで、使用可能なすべての Java ラ イブラリー・ファイルとメソッドが所属するクラスのクラス・ファイルを検索しま す。

Java ライブラリーまたはクラスの配置は、クラスを含むライブラリー・ファイルま たはクラス・ファイルを、マクロ・ランタイムがマクロの再生中に検出できる場所 に配置することから構成されます。

以下のタイプの Java クラスを配置する必要はありません。

- Java API のクラス (Java アーカイブ・ファイルがクライアント・ワークステーション上にすでに存在して、その位置が Host On-Demand クライアントの起動時に指定されるクラスパスにリストされている)。
- Host On-Demand マクロ・ユーティリティー・ライブラリー (HML ライブラリ ー)のクラスが、Host On-Demand クライアント・コードと共に保管されてい る (173 ページの『マクロ・ユーティリティー・ライブラリー (HML ライブラ リー)』を参照)。

マクロ・スクリプトによって呼び出されるメソッドを含むその他の Java クラス は、マクロ・ランタイムが検出できる場所にユーザーが配置する必要があります。 環境に応じて、クラス・ファイルとして、または Java クラスを含むライブラリー として Java クラスを配置できます。

Java ライブラリーとクラスの配置について詳しくは、「*Host On-Demand* 計画、イ ンストール、および構成」の『カスタマー提供の Java アーカイブおよび Java クラ スのデプロイ (Deploying customer-supplied Java archives and classes)』を参照 してください。

変数名と型名

変数名の規則は次のとおりです。

- 変数名には、英数字 (a から z、A から Z、0 から 9)、下線 (_)、またはハイフン (-) のみを含めることができます。
- 大/小文字の区別があります (例えば、 strTmp と strtmp は 2 つの異なる名前 です)。
- 変数名は、インポート型の短縮名または完全修飾クラス名と同じであってはなりません。

型名の規則は次のとおりです。

- 型名には、英数字、下線 (_)、ハイフン (-)、またはピリオド (.) のみを含めるこ とができます。
- 大/小文字の区別があります。

マクロ間での変数の転送

1 つのマクロを別のマクロに「チェーニング」する (戻りのない呼び出し) PlayMacro アクションの場合は、呼び出し側マクロのすべての変数とその値をター ゲット・マクロに転送できます。ターゲット・マクロは、自身の変数と転送された 変数の両方にアクセスできます (109 ページの『PlayMacro アクション (<playmacro> エレメント)』を参照)。

フィールド変数

フィールド変数は、ストリング変数の一種です。フィールド変数は、ストリング変 数と同様にストリングを格納し、ストリング変数が有効であるすべてのコンテキス トでフィールド変数を使用できます。

ただし、フィールド変数にストリングが格納される方法について、フィールド変数 とストリング変数は異なっています。フィールド変数に格納されるストリングは常 に、マクロ・ランタイムが現行セッション・ウィンドウ内の 3270 または 5250 フ ィールドから読み取るストリングです。マクロ・ランタイムがこのストリングを 3270 または 5250 フィールドから読み取るようにするには、次の項目を指定して変 数更新アクションを作成する必要があります。

- 1. フィールド変数の名前 (\$fldFilename\$ など)。
- 2. 位置ストリング ('5,11' のように、コンマで区切った 1 対の整数を含むストリ ング)。

マクロ・ランタイムは、変数更新アクションを実行するときに次のステップを行い ます。

- 1. セッション・ウィンドウ内で、位置ストリングに指定された行と列の位置を検索 する。
- 2. その行と列の値が位置指定している 3270 または 5250 フィールドを検索する。
- 3. フィールドの内容全体を読み取る。
- 4. フィールドの内容全体をストリングとしてフィールド変数に格納する。

詳しくは、 132 ページの『フィールド変数に対する変数更新アクション』を参照し てください。

変数の使用

変数が初期化される時点

マクロ・ランタイムは、マクロの再生を開始するとき、マクロ画面を処理する前に 変数に初期値を割り当てます。

標準型に属する変数の使用

変数が保持する値の使用

標準型 (string、integer、double、boolean) に属する変数は、同じ型の即時値 ('Elm Street'、10、4.6e-2、true など) とほぼ同じ方法で使用できます。

- このサブセクションに後でリストする制限を例外として、標準型の変数は、同じ データ型の即時値を使用できる任意の入力フィールド (マクロ・エディターの場 合) または属性 (コード・エディターの場合) で使用できます。例えば、入力フ ィールド (「メッセージ・アクション (Message action)」ウィンドウの「メッセ ージ・テキスト (Message Text)」フィールドなど) がストリング値を必要とする 場合は、フィールドも同様にストリング変数を受け入れます。 41 ページの『等 価』を参照してください。
- 変数は、同じ型の即時値と同じ使用方法で、演算子や式と組み合わせて使用できます。 37 ページの『演算子および式』を参照してください。
- 変数の型と異なるコンテキストにある変数の値は、同じ型の即時値が変換される 場合と同じ方法で、適切な型の値に変換されます (可能な場合)。 40 ページの 『自動データ型変換』を参照してください。

ただし、特定のコンテキストでは変数を使用できません。マクロ・エディターで は、次のコンテキストで変数を使用できません。

- 「一般 (General)」タブのフィールドすべて。
- 「画面 (Screens)」タブの「画面名 (Screen Name)」フィールド。

 「PlayMacro アクション (PlayMacro action)」ウィンドウのフィールドすべての 値。

コード・エディターでは、次のコンテキストで変数を使用できません。

- エレメントすべての属性の名前。
- <HAScript> エレメントの属性すべての値。
- <screen> エレメントの name 属性の値。
- <description> エレメントの uselogic 属性の値。
- <nextscreen> エレメントのマクロ画面の名前。
- <playmacro> エレメントの属性すべての値。

標準型に属する変数への値の書き込み

標準型に属する変数には、次の方法で値を書き込むことができます。

- 変数の作成時に初期値を割り当てる。
- 変数更新アクションを使用して変数に値を割り当てる。
- プロンプト・アクションを使用してユーザー入力を取得し、変数に割り当てる。
- 抽出アクションを使用してセッション・ウィンドウからデータを読み取り、変数 に割り当てる。
- ・ 戻りコード値を変数に書き込むアクションを使用する(プログラム実行アクションや印刷アクションなど)。

制限: 標準型の変数に、次の値のいずれかを割り当てることはできません。

- 値 null。(例外: 値 null をストリング変数に割り当てた場合、値はストリング 'null' に変換されます)
- void メソッドの呼び出し。
- 配列を戻すメソッドの呼び出し。

標準型の変数に Java オブジェクトを書き込む: 標準型の変数に Java オブジェクトを書き込むと、マクロ・ランタイムはインポート型の toString() メソッドを呼び 出してから、結果ストリングを変数に割り当てることを試みます。

インポート型に属する変数の使用

変数が保持する値の使用

インポート型に属する変数に格納されている値は、次のように使用できます。

- 変数更新アクションを使用して、同じ型の別の変数に変数を割り当てることができます。
- ・ 変数に対して Java メソッドを呼び出すことができます(172 ページの『Java メソッドの呼び出し』を参照)。Java メソッドが標準型 (string、integer、 double、boolean) に属する値を戻した場合は、その型の値を使用する場合と同じ ように結果を使用できます。

制限

次の型のデータは、インポート型の変数に割り当てることはできません。

• 標準型 (string、integer、double、boolean、field) に属する値または変数。

- 別のインポート型のインスタンス、またはその型に属する変数 (その型がインポート型のスーパークラスである場合を除く)。
- インポート型の変数に対して呼び出したメソッドから戻されたオブジェクトのインスタンスの配列。

インポート型の変数にマクロがこれらの無効な型の値を割り当てようとすると、マ クロ・ランタイムはランタイム・エラーを生成し、マクロを停止します。

インポート型に属する変数への書き込み

インポート型の変数には、次の方法で値を書き込むことができます。

- 変数の作成時に、変数に値を割り当てることができます。
- 変数更新アクションを使用して、変数に値を割り当てることができます。

インポート型に属する変数には、次の型の変数を割り当てることができます。

- 同じ型のインスタンス。このインスタンスは、同じ型の変数のもの、または同じ 型のインスタンスを戻すメソッドの呼び出しから得られたもののどちらかです。
- 値 null。値 null を指定するには、次のどちらかを使用できます。
 - キーワード null。
 - 「変数 (Variables)」タブの「初期値 (Initial Value)」フィールド、「変数更新 (Variable update)」ウィンドウの「値 (Value)」フィールドなどのブラン ク入力フィールド (マクロ・エディターを使用している場合)。
 - 次に示す <create> エレメントの value の属性のような空の属性 (コード・ エディターを使用している場合)。

<create name=\$ht\$ type="Hashtable" value="" />

同じインポート型の変数の比較

同じインポート型の 2 つの変数を比較する条件式 (例えば、条件アクションの「条件 (Condition)」フィールド内)の中では、変数自体を使用するのではなく、基礎クラスにある比較メソッド (equals() など) をインプリメントする必要があります。 例えば、次のとおりです。

\$htUserData.equals(\$htPortData\$)\$

代わりに、変数自体を比較すると (例: \$htUserData\$ == \$htPortData\$)、次のよう になります。

- 1. マクロ・ランタイムは、それぞれの変数ごとに、基礎 Java クラスの toString() メソッドを呼び出し、ストリングの結果を得る。
- 2. マクロ・ランタイムは、2 つのストリングの結果を比較し、ブールの結果を得る。
- 3. マクロ・ランタイムは、条件の結果をステップ 2 で得られたブールの結果に設 定する。

これにより得られる結果は、2 つの変数の比較結果としては正しくない可能性があります。

パラメーター・リストのマクロへの引き渡し

通常の言語では、パラメーター・リストは、プログラムが動作するように指示し、 エンド・ユーザーがプログラムを起動したときに指定する値の集合です。例えば、 copy がファイルを現行ディレクトリーから別のディレクトリーにコピーするプログ ラムの名前なら、次のコマンド行の呼び出し

copy MyFile.txt c:¥temp

により、コピー・プログラムを名前で呼び出して (copy)、コピー対象のファイル名 (MyFile.txt)、および宛先ディレクトリー (c:¥temp) を含むパラメーター・リスト (MyFile.txt c:¥temp) を指定します。

Host On-Demand には、プログラムが呼び出されたときにパラメーター・リストを プログラムに引き渡すことに類似したマクロ機能があります。マクロを起動する 前、またはマクロの起動と同時に、エンド・ユーザーは名前値のペアのリストを指 定できます。各ペアは、マクロの内部で定義された変数、およびマクロ・ランタイ ムが変数に割り当てる必要のある初期値で構成されています。

例えば、マクロ FileDownload がファイルをリモート・ホストからローカル・ワー クステーションにダウンロードし、このマクロに変数 strRemoteFile と strLocalFile が含まれる場合、エンド・ユーザーは以下のパラメーター・リストを 指定することができます。

strRemoteFile="NewData.123", strLocalFile="MyData.123"

エンド・ユーザーがマクロを再生するとき、マクロ・ランタイムは、パラメータ ー・リストで指定された各変数を、パラメーター・リストで指定された値に初期設 定します。マクロは、ファイル NewData.123 をリモート・ホストからダウンロード して、そのファイルを MyData.123 に名前変更します。

もちろん、上記の手順どおりに処理するには、マクロの作成者は、必要な情報を変数 strRemoteFile と strLocalFile 内で検索するようマクロをコード化する必要があります。

一般的に、マクロの作成者は以下の条件を満たす必要があります。

- パラメーター・リストで使用される変数を作成する。
- これらの変数内で、マクロの起動時に必要な情報をマクロに検索させる。

また、良いプログラムを書くために、マクロの作成者は以下の動作に注意する必要 があります。

- マクロ記述内 (マクロ・エディターの「マクロ (Macro)」タブの「記述 (Description)」入力フィールド) でのパラメーター・リストの記述。
- 変数が不適切に初期化された場合のエンド・ユーザーへのメッセージの表示。

詳しくは、 171 ページの『マクロ作成者の考慮事項』を参照してください。

エンド・ユーザーは以下の条件を満たす必要があります。

• パラメーター・リストを正しく指定する。

ただし、システム管理者は、パラメーター・リストの指定作業を、以下のいずれか でパラメーター・リストを事前定義することによって処理することができます。

- セッション構成 (セッション開始時にマクロが自動的に開始する場合)、または
- クリック時にパラメーター・リストを使用してマクロを再生するカスタマイズされたボタン (メイン・ツールバーで作成)。

詳しくは、『パラメーター・リストの指定』を参照してください。

Host On-Demand によるパラメーター・リストの処理方法

マクロ内のすべての変数には、パラメーター・リストで使用されているかどうかに かかわらず、初期値があります。この初期値は、デフォルトの初期値(156 ページ の表 22 を参照)か、マクロの作成者が割り当てた初期値です。

パラメーター・リストは関連するマクロの外部に存在します。マクロ自身の内部に 保管されるのではなく、マクロの再生が開始されたときに、マクロ・ランタイムに 受け渡されます。

マクロ・ランタイムがマクロを再生する準備ができると、まず最初にマクロの各変 数をその変数に対してマクロ自身内で定義された初期値に初期設定します。

次に、マクロ・ランタイムはパラメーター・リストを検索します。パラメーター・ リストが見つかると、マクロ・ランタイムはパラメーター・リストのそれぞれの名 前値のペアを処理して、指定された各変数をリストで指定された値に再初期設定し ます。

マクロ・ランタイムはマクロの再生を通常の方法で継続します。

パラメーター・リストの指定

パラメーター・リストの指定場所

表 23 には、マクロのパラメーター・リストを指定できる Host On-Demand ユー ザー・インターフェースの場所が記載されています。

ユーザー・インターフェース	パラメーター・リストを含む	Host On-Demand ガ
の領域:	入力フィールドへの	パラメーター・リストを
	ナビゲート方法:	保管するか ?
セッション構成	 「開始オプション (Start Options)」ウィンドウ 「自動開始オプション (Auto-Start Options)」グ ループ 	はい。 入力リストはセッション構成 の一部として保管されます。
	– 「パラメーター (Parameters)」入力フ ィールド	

表 23. マクロのパラメーター・リストの指定

表 23. マクロのパラメーター・リストの指定 (続き)

ユーザー・インターフェース	パラメーター・リストを含む	Host On-Demand \hbar^3
の領域・	スカフィールドへの	パラメーター・リストを
· > PR->1.	ナビゲート方法・	保管するか?
メイン・ツールバー 注意:メイン・ツールバー は、ツールバー・オブジェク トとして保管することがで き、他のセッション構成にイ ンポートすることができま す。	「編集 (Edit)」>「設定 (Preferences)」>「ツールバー (Toolbar)」>「追加 (Add)」 ボタン • 「マクロ (Macro)」ウィン ドウ - 「パラメーター (Parameters)」入力フ ィールド	はい。 パラメーター・リストはメイ ン・ツールバーの一部として 保管されます。
「マクロを再生 (Play macro)」ダイアログ	「操作 (Actions)」>「マクロ を再生 (Play Macro)」 • 「パラメーター (Parameters)」入力フィー ルド	いいえ。 パラメーター・リストは、以 下のいずれかで別の選択が実 行されるまで、一時的に「パ ラメーター (Parameters)」入 カフィールドへ保管されま す。 ・ 「マクロを再生 (Play macro)」ダイアログ。ま たは ・ 「選択可能なマクロ (Available macros)」ダイ アログ
「選択可能なマクロ (Available macros)」ダイア ログ	マクロ・マネージャー・ツー ルバーの「マクロを選択 (Select a macro)」アイコン ・ 「パラメーター (Parameters)」入力フィー ルド	いいえ。 パラメーター・リストは、以 下のいずれかで別の選択が実 行されるまで、一時的に「パ ラメーター (Parameters)」入 力フィールドへ保管されま す。 ・ 「マクロを再生 (Play macro)」ダイアログ。ま たは ・ 「選択可能なマクロ (Available macros)」ダイ アログ
表 23. マクロのパラメーター・リストの指定 (続き)

ユーザー・インターフェース	パラメーター・リストを含む	Host On-Demand が
の領域:	入力フィールドへの	パラメーター・リストを
	ナビゲート方法:	保管するか?
ポップアップ・キーパッド・	メイン・ウィンドウで「編	はい。
ダイアログのカスタマイズ	集」>「設定」>「ボッブアッ プ・キーパッド」>「カスタ	パラメーター・リストはポップアップ・キーパッドの一部
	マイズ (Customize)」また は、ポップアップ・キーパッ	として保管されます。
	ド・ウィンドウで「編集」> 「カフタマイブ	
	(Customize)」を選択しま す。	

ただし、現在選択しているマクロを、マクロ・マネージャー・ツールバーの「マク ロを再生 (Play Macro)」アイコンを使用して再生している場合、パラメーター・リ ストを指定することはできません。代わりに、「選択可能なマクロ (Available macros)」ダイアログ (上の表の行 4 を参照) でパラメーター・リストを指定して、 現在選択しているマクロを再生します。

上の表では、Host On-Demand は、パラメーター・リストを表の最初の 2 行で指 定された場所に対してのみ保管することに注意してください (セッション構成の自 動開始セクションと、メイン・ツールバーに追加されたカスタマイズされたボタ ン)。

また、Host On-Demand は、パラメーター・リストをマクロ自身の一部としてでは なく、(セッション構成の一部として、またはメイン・ツールバーの一部として)マ クロの外部に保管することに注意してください。

パラメーター・リストの形式

パラメーター・リストを指定する場合は (例えば、 165 ページの表 23 の行 2 で 示されているように、「ボタンを追加 (Add Button)」ダイアログの「マクロ (Macro)」ウィンドウの「パラメーター (Parameters)」入力フィールドで指定)、以 下の形式で指定する必要があります。

name1="value1", name2="value2", name3="value3", ...

例えば、次のとおりです。

strRemoteFile="NewData.123", strLocalFile="MyData.123"

形式の規則は以下のとおりです。

• パラメーター・リスト

パラメーター・リストは名前と値のペアの集合です。各ペアがマクロで定義され た変数を参照している限り、指定できるペアの数に制限はありません。

名前値のペアを区切るには、ブランク・スペース (「」)、コンマ (「,」)、また はその両方 (「,」) を使用します。

名前値のペア

名前値のペアで変数名を指定する場合には、等号(「=」)を後に続け、その後に 二重引用符("")で囲まれた値を続けます。例えば、次のとおりです。 strLocalFile="MyData.123"

変数名

変数名はマクロで記述されているスペルどおりに入力します。大/小文字の区別 に注意してください。例えば、strLocalFile と strLocalfile は異なります。

変数名を指定する場合は、ドル記号 (\$) で囲んでも囲まなくてもかまいません。 正 : \$strLocalFile\$。正 : strLocalFile。

変数はプリミティブ変数、つまり、標準型 (integer、double、string、または boolean) に属している変数である必要があります。(インポートされた Java ク ラスに基づく) インポート型に属している変数を指定することはできません。

• 等号 (「=」)

等号は変数名と値の間に必要です。

変数値

変数値は二重引用符 ("") で囲む必要があります。

値は単純な値で、式にしないようにする必要があります。

値が変数以外の異なる標準データ型に属している場合、Host On-Demand は通 常の方法で自動的に値を正しいデータ型に変換しようとします (40 ページの 『自動データ型変換』を参照)。

ストリング変数の規則

- ストリングは単一引用符で囲まない。誤:strAddress="'123 Elm Street'"。
 正:strAddress="123 Elm Street"。
- 空ストリングを指定するには、間に何もない 2 つの二重引用符を使用する。
 空ストリング: ""。
- 単一引用符 (') と円記号 (¥) にはエスケープ・シーケンスを使用する。
 - 単一引用符を指定するには、¥'を使用する。例: "John¥'s Business Plan"。
 - 円記号を指定するには、¥¥ を使用する。例: "c:¥¥Documents and Settings"。

doubles、integers、booleans の規則

 - 余分なスペースを値の周りに含むことができる。Host On-Demand は余分な スペースを破棄します。例:"10"、"423.25 "、"true "。

各プリミティブ変数型の例は以下のとおりです。

```
intLineCount="24", dblLength="1441.25", strName="John Smith", boolComplete="true"
```

パラメーター・リストでのエラーの確認

Host On-Demand は、エンド・ユーザーがマクロを再生するまで、パラメーター・ リストでエラーをチェックしません。エンド・ユーザーがマクロを再生するとき、 Host On-Demand がパラメーター・リストでエラーを検出すると、以下のことを実 行します。

- マクロの再生を終了する。
- エンド・ユーザーに対してエラー・ダイアログ・ボックスを表示するか、メッセ ージを Java コンソールに書き込んで、エラー・メッセージを生成する。

Host On-Demand は、3 つのタイプのエラー状態を検査します。

• 指定された変数がマクロに存在しない。

このタイプのエラーでは、マクロに存在しないか、または間違って入力されている (例えば、strLocalFile ではなく strLocalfile) 変数名がパラメーター・リ ストに含まれています。

このタイプの問題を回避するための処置は、以下のとおりです。

- エンド・ユーザーに変数名を正しく入力するよう警告する。または
- メイン・ツールバーで、正しいパラメーター・リストを使用したマクロを再 生するカスタマイズされたボタンを作成する。
- 構文エラーがパラメーター・リストに存在する。

このタイプのエラーでは、パラメーター・リストに構文エラーが存在します (例 えば、値が二重引用符で囲まれていません)。

このタイプの問題を回避するための処置は、以下のとおりです。

- エンド・ユーザーに構文規則を守るように警告する。または
- メイン・ツールバーで、正しいパラメーター・リストを使用したマクロを再 生するカスタマイズされたボタンを作成する。

170 ページの図 53 には、構文エラーの例と、Host On-Demand のエラーの処 理方法が記載されています。

strRemoteFile="NewData.123 intLength="14622"

Host On-Demand:

- 1) Sets the variable strRemoteFile to "NewData.123 intLength=".
- 2) Tries to process 14622" but cannot (Host On-Demand is looking for a variable name but instead finds an integer followed by a double quote).
- 3) Terminates the macro playback and generates an error message.

strRemoteFile="NewData.123"98 strLocalFile="MyData.123"

Host On-Demand:

1) Sets the variable strRemoteFile to "NewData.123".

- 2) Tries to process **98** but cannot
- (Host On-Demand is looking for a variable name but finds an integer).
- 3) Terminates the macro playback and generates an error message.

strRemoteFile=NewData.123 strLocalFile=MyData.123 Host On-Demand:

- 1) Does not find a double quote after the first equals sign (=).
- Terminates the macro playback and generates an error message.

図 53. 構文エラーの例

ランタイム・エラーが発生する。

ランタイム・エラーの原因となるパラメーター・リスト内の問題の例は、以下の とおりです。

intLength="abc"

上の例では、変数 intLength は integer 型です。 Host On-Demand は、スト リング abc を 10 進法の整数に変換しようとして失敗します。

ランタイム・エラーが発生すると、Host On-Demand は以下のことを実行しま す。

- マクロの再生を終了する。
- エラー・メッセージを含むエラー・ダイアログ・ボックスを表示する。

上のランタイム・エラーの例では、Host On-Demand は、以下のようなメッセ ージを含むエラー・ダイアログ・ボックスを表示します。「変数 \$intLength\$ を更新中に、型のミスマッチが発生しました。」

パラメーター・リストのプログラムからの指定

Host Access Toolkit (Host On-Demand CD-ROM に収録) には、パラメーター・ リストを管理する 2 つの Java メソッドがあります。

public Properties getPrimitiveMacroVariables()

マクロの各プリミティブ変数に対して、変数名とマクロが変数を設定する初期値 を戻します。このメソッドを使用して、パラメーター・リストに含めるマクロの すべての変数のリストを取得します。

public void setVariableParameters (Properties p)

変数名と各パラメーターの値で構成されているマクロの入力パラメーター・リス トを指定します。

詳しくは、Host Access Toolkit に同梱されている文書を参照してください。

マクロ作成者の考慮事項

マクロをコード化してパラメーター・リストを使用するとき、以下のことに注意し てください。

- マクロの作成者は、マクロの内部にスイッチを設定して、特定の変数がパラメー ター・リストを使用して初期化されるようにする必要はなく、また、初期化する ことはできない。(もちろん、マクロの作成者は、この目的でコメントを追加する ことはできます。)パラメーター・リストに関する情報はマクロ内部に保管され ません。
- パラメーター・リストが存在しなくても、各変数には初期値があり、その初期値 はデフォルトの初期値(156ページの表 22 を参照)か、マクロの作成者が指定 した初期値である。

ー般的な原則として、エンド・ユーザーが構文エラーなどのエラーを発生させず、 パラメーター・リストを正しく指定できる必要があることに注意してください。エ ンド・ユーザーの助けになるように、以下の処置に注意してください。

- マクロ・エディターの「マクロ (Macro)」タブの「記述 (Description)」フィール ドでパラメーター・リストを記述する。エンド・ユーザーは、以下のいずれかか らマクロの「プロパティー」ダイアログを表示して、「記述 (Description)」フィ ールドの内容を表示することができます。
 - 「選択可能なマクロ (Available macros)」ダイアログ (マクロ・マネージャ ー・ツールバーの「マクロを選択 (Select a macro)」アイコンをクリック)。
 - 「マクロを再生 (Play macros)」ダイアログ (「操作 (Actions)」>「マクロを 再生 (Play macro)」をクリック)。
- 不適切に初期化された変数を検査するようにマクロをコード化して、変数が適切 に初期化されない場合は、警告メッセージを通知する。
- メイン・ツールバーで、正しいパラメーターを使用したマクロを再生するカスタ マイズされたボタンを作成する。

チェーニングされたマクロで変数を初期化しないパラメーター・リ スト

パラメーター・リストは、エンド・ユーザーがパラメーター・リストを使用して起 動するマクロ内でのみ変数を初期化します。パラメーター・リストは、先に出現す るマクロがチェーニングする後続のマクロ内の同じ名前の変数を初期化しません。 (チェーニングされたマクロに関する情報について、詳しくは 109 ページの 『PlayMacro アクション (<playmacro> エレメント)』を参照してください)。

例えば、以下の状態があるとします。

- エンド・ユーザーがパラメーター・リスト strRemoteFile="NewData.123" を使用したマクロ FileDownload を開始する。
- マクロ FileDownload は後でマクロ Download01 にチェーニングする。
- マクロ Download01 も strRemoteFile という変数を含んでいる。

この状態では、Host On-Demand は、チェーニングされたマクロの再生を準備する 際に、チェーニングされたマクロ内の変数 strRemoteFile を NewData.123 に初期 化しません。代わりに、Host On-Demand は、チェーニングされたマクロに何らか の初期値 (デフォルトの初期値かマクロの作成者が指定した初期値) が指定されてい れば、通常通りチェーニングされたマクロ内の変数を初期化します。

上の状態と、マクロがチェーニングしているときに発生する変数の転送を混同しな いでください。マクロ A がマクロ B にチェーニングしている場合には、マクロ A はすべての変数と現行値を B に受け渡すことができます (110 ページの『変数の 転送』を参照)。この機能を利用して、値を呼び出し側のマクロからチェーニングさ れたマクロに受け渡すことができます。

Java メソッドの呼び出し

メソッド呼び出しを使用できる個所

メソッドから戻される値が有効であるすべてのコンテキストで、メソッドを呼び出 すことができます。例えば次のように、入力アクションの中で行の値をメソッドか ら戻される整数値に設定できます。

\$importedVar.calculateRow()\$

また、メソッドの戻り変数が必要ない場合や、メソッドに戻り値がない (void) 場合 は、実行アクションを使用してメソッドを呼び出すこともできます (107 ページの 『実行アクション (<perform> エレメント)』を参照)。

メソッド呼び出しの構文

インポートしたクラスに属するメソッドを呼び出すには、Java で使用するものと同 じ構文を使用します。ただしその構文に加え、変数の場合と同じように、メソッド 呼び出しをドル記号 (\$) で囲む必要があります。例えば、次のとおりです。

\$new FileInputStream('myFileName')\$
\$fis.read()\$

メソッドにパラメーターとして渡す即時ストリング値 ('Elm Street' や上の最初の 例の 'myFileName' など) は、通常どおり単一引用符で囲む必要があります (34 ペ ージの『拡張マクロ形式のストリング表記規則』を参照)。

マクロ・ランタイムが呼び出し先メソッドを検索する方法

メソッド呼び出し (\$prp.get('Group Name')\$ など) をマクロ・スクリプトに追加す る際に、マクロ・エディターは、呼び出し先メソッドまたはコンストラクターが変 数の属するクラスに存在するかどうかを検査しません。その検査は、呼び出しが行 われたときにマクロ・ランタイムによって行われます。

メソッドは、基礎 Java クラスの public メソッドでなければなりません。

マクロ・ランタイムは、ユーザーが呼び出したメソッドに一致するメソッドがある かどうか Java クラスを検索する際に、 173 ページの表 24 に示すようにマクロの データ型 (boolean、integer、string、field、double、インポート型) を Java デー タ型にマップします。

表 24. マクロ・ランタイムがマクロのデータ型を Java データ型にマップする方法

メソッド・パラメーターが以下のマクロ・	マクロ・ランタイムは以下の Java データ型
データ型に属する場合:	のパラメーターをもつ Java メソッドを検索
	する:
boolean	boolean
integer	int
string	ストリング (String)
field	ストリング (String)
double	double
インポート型	インポート型の基礎クラス

マクロ・ランタイムは、呼び出し先メソッドを次のように検索します。

- 1. マクロ・ランタイムは、インポート型の定義に指定されているクラスを検索する (java.util.Properties など)。
- マクロ・ランタイムは、そのクラス内で呼び出し先メソッドと同じメソッド・シ グニチャー (名前、パラメーター数、およびパラメーターの型)をもつメソッド を検索する。
- 3. 検索が成功した場合、マクロ・ランタイムはメソッドを呼び出す。
- 検索が失敗した場合、マクロ・ランタイムはそのクラス内で、呼び出し先メソッドと同じ名前とパラメーター数 (パラメーターの型は無視)をもつメソッドを検索する。
 - a. マクロ・ランタイムが該当するメソッドを検出した場合は、指定されたパラ メーターを使用してメソッドを呼び出す。
 - b. 呼び出しがエラーを出さずに戻った場合、マクロ・ランタイムは正しいメソ ッドを呼び出したものと想定する。
 - c. 呼び出しがエラーを出して戻った場合、マクロ・ランタイムは別のメソッド を検索する。
 - d. 同じ名前とパラメーター数をもつメソッドをすべて試すまで、検索が継続される。どれも成功しなかった場合、マクロ・ランタイムはランタイム・エラーを生成する。

マクロ・ユーティリティー・ライブラリー (HML ライブラリー)

Host On-Demand マクロ・ユーティリティー・ライブラリー (HML ライブラリー) は、 Host On-Demand クライアント・コードとともにパッケージされているユー ティリティー・ライブラリーです。以下の項目を実行せずに、これらのライブラリ ーのうちの 1 つからメソッドを呼び出すことができます。

- 基礎クラスのインポート。または
- クラスのインスタンスを含むための変数の作成。または
- クラスのインスタンスの作成

実際には、HML Java ライブラリーに含まれているクラスのインポート、HML クラスに属する変数の作成、または HML オブジェクトのインスタンスの作成は許可 されていません。 これは、マクロの再生が開始されたときに進む初期化の実行中に、マクロ・ランタ イムが以下の処理を実行するためです。

- すべての HML クラスをインポートする。
- 各 HML クラスに対して、クラスのインスタンスを含む 1 つの変数を作成す る。
- 各 HML クラスのインスタンスを 1 つ作成して、適切な変数に格納する。

以下の表には、各 HML 変数に対する変数名と基礎クラスのメソッドの型が記載されています。

表 25. HML 変数

HML 変数:	メソッドの説明 :
\$HMLFormatUtil\$	ストリングのフォーマット設定のためのメソッド。
\$HMLPSUtil\$	セッション・ウィンドウの表示スペースにアクセスするメ
	ソッド。
\$HMLSessionUtil\$	セッション値を戻すメソッド。
\$HMLSQLUtil\$	最新の SQLQuery アクションの結果に関する情報を戻す
	メソッド
\$HMLSystemUtil\$	アプレット、Java プロパティー、および OS 環境に関す
	る情報を戻すメソッド。

HML ライブラリーに属するメソッドの呼び出し

HML ライブラリーに属するメソッドを呼び出すには、通常の方法で、変数名、メ ソッド名、入力パラメーターを指定します。

```
$HMLFormatUtil.numberToString(1.44)$
$HMLPSUtil.getCursorPos()$
$HMLSessionUtil.getHost()$
```

図 54. HML メソッドの呼び出し例

HML で開始される予約済み変数名

通常の変数と HML 変数の混同を回避するために、HML で開始される変数名は予約 済みです。HML で開始される変数を作成しようとすると、Host On-Demand はエラ ー・メッセージを生成します。

\$HMLFormatUtil\$

\$HMLFormatUtil\$ を使用して呼び出されるメソッドは、フォーマット設定メソッド です。 175 ページの表 26 には、これらのメソッドの要約が記載されています。

表 26. \$HMLFormatUtil()\$ のメソッドの要約

メソッドの要約 : \$HMLFormatUtil\$		
ストリング (String)	numberToString(Object obj) 数値を現在構成されているロケールに従って、フォーマット設定し たストリングに変換します。入力パラメーターの型は、integer ま たは double です。	
int または double	<pre>stringToNumber(String str)</pre>	

数値と現行ロケール形式の相互変換

ロケールは、各国語と地域に関連したフォーマット設定の規則の集合です。例え ば、クライアント・ワークステーションが構成されているロケールに従って、 1111.22 などの 10 進法の値が以下のストリングのいずれかで表されます。

```
'1111.22'
'1,111.22'
'1111,22'
```

一方、-78 のような負の数値の場合は、以下のように表されます。 '-78'

```
'78-'
```

メソッド numberToString() および stringToNumber() を使用すると、数値 (1111.22 などの、integer 型または double 型の変数や即時値) と現行のロケールで の表記 (「1111.22」、「1,111.22」、または「1111,22」などのストリング) との変 換を実行できます。

メソッドの詳細

numberToString():

public String numberToString(Object obj)

このメソッドは、数値 (integer または double) を現在構成されているロケールに 従ってフォーマット設定したストリングに変換します。入力パラメーターの型は、 integer または double です。

このメソッドは、推奨されていない独立したメソッド \$FormatNumberToString()\$ に置き換わるものです。

```
<input value="$HMLFormatUtil.numberToString(1111.44)$"
row="20" col="16" movecursor="true"
xlatehostkeys="true" encrypted="false" />
```

stringToNumber():

```
public int stringToNumber(String str)
public double stringToNumber(String str)
```

このメソッドは、現在構成されているロケールに従ってフォーマット設定した数値 ストリングを数値に変換します。戻された数値は、入力ストリングに応じて、 integer 型か double 型になります。

このメソッドは、推奨されていない独立したメソッド \$FormatStringToNumber()\$ に置き換わるものです。

```
<message title="'stringToNumber()'" value="'1111.33'" />
<extract name="'Extract'" planetype="TEXT_PLANE"
srow="1" scol="1"
erow="1" ecol="10" unwrap="false"
assigntovar="$value$" />
<if condition="$HMLFormatUtil.stringToNumber($value$)$ < 0 "
....
```

図 56. stringToNumber() の例

\$HMLPSUtil\$

\$HMLPSUtil\$ を使用して呼び出されたメソッドは、セッション・ウィンドウの表示 スペースに影響を与えるか、セッション・ウィンドウの表示スペースに関する情報 を戻します。表 27 には、これらのメソッドの要約が記載されています。

表 27. \$HMLPSUtil\$ のメソッドの要約

メソッドの要約 : \$HMLPSUtil\$		
int	convertPosToCol(int pos)	
	表示スペースの指定された位置の列番号を戻します。	
int	convertPosToRow(int Pos)	
	表示スペースの指定された位置の行番号を戻します。	
void	enableRoundTrip(boolean flag)	
	双方向言語の場合、双方向文字の後に続く数表示が、双方向文字と	
	場所を交換するかどうかを決定します。	
int	getCursorCol()	
	表示スペースのテキスト・カーソルの列番号を戻します。	
int	getCursorPos()	
	表示スペースのテキスト・カーソルの位置を戻します。	
int	getCursorRow()	
	表示スペースのテキスト・カーソルの行番号を戻します。	
int	getSize()	
	表示スペースのサイズを戻します (表示スペースの文字位置の数)。	

表 27. \$HMLPSUtil\$ のメソッドの要約 (続き)

メソッドの要約	メソッドの要約 : \$HMLPSUtil\$		
int	getSizeCols() 表示スペースの列数を戻します。		
int	getSizeRows() 表示スペースの行数を戻します。		
ストリング (String)	getString(int pos, int len) 表示スペースの指定された位置から開始される、指定された長さの テキスト・ストリングを戻します。		
int	searchString(String str) 指定されたストリングの表示スペースの位置を戻します (指定され たストリングが表示スペース内に検出されない場合には、0 になり ます)。		

表示スペース

表示スペースは、セッション・ウィンドウ内に各行と列の位置のエレメントを含む データ構造です (ただし、セッション・ウィンドウの最後の行は、オペレーター情 報域に使用されるため、含まれません)。表示スペースのサイズは、セッション・ウ ィンドウのサイズによって決定されます。例えば、セッション・ウィンドウが 80 行と 25 列の場合、表示スペースのサイズは 80 * 24 = 1920 になります。

表示スペースのエレメントの位置は、セッション・ウィンドウの行と列の位置に連 続的に対応しており、左から右、上部から下部へと読み取ります。例えば、セッシ ョン・ウィンドウが 80 行と 25 列の場合、以下の表のようになります。

Row of Session Window: 1	Column of Session Window: 1	Corresponds to element at this position in PS: 1
1	2	2
1	3	3
•••		
1	80	80
2	1	81
2	2	82
2	3	83
24 24	79 80	1919 1920

図 57. セッション・ウィンドウの行と列の位置と表示スペースでの位置の対応

Host On-Demand は表示スペースを使用して、セッション・ウィンドウに表示され る文字を格納します。表示スペースの各エレメントは、1 文字 (および輝度などの 文字に関する情報) を格納するために使用されます。例えば、ストリング Message がセッション・ウィンドウの行 1 と列 1 に表示されると、以下の図のようになり ます。

Column of Session Window:	Corresponds to element at this pos- ition in PS:	Character stored in this element:
1	1	Μ
2	2	е
3	3	S
4	4	S
5	5	a
6	6	g
7	7	e
	Column of Session Window: 1 2 3 4 5 6 7	Column of Session Window:Corresponds to element at this pos- ition in PS:11223344556677

図 58. 「Message」が行1と列1に表示される場合のレイアウト

表 28 には、さまざまな値を計算するための公式 (通常使用する必要はありません) が記載されています。これらの公式で使用されている記号の意味は以下のとおりで す。

- row セッション・ウィンドウの行の位置
- col セッション・ウィンドウの列の位置
- pos 表示スペース内での位置
- NUMROWS セッション・ウィンドウの行数 (オペレーター情報域 (OIA) で使用さ れる最後の行は含まない)
- NUMCOLS セッション・ウィンドウの列数

表 28. 表示スペースに関連した値を計算する公式

値	計算のための公式	
表示スペースのサイズ	NUMROWS * NUMCOLS	
	Example: 24 * 80 = 1920	
row	(pos + NUMCOLS - 1) / NUMCOLS	
	Example: (81 + 80 - 1) / 80 = 2	
col	pos - ((row - 1) * NUMCOLS)	
	Example: 1920 - ((24 - 1) * 80) = 80	
pos	((row - 1) * NUMCOLS) + col	
	Example: ((24 - 1) * 80) + 1 = 1641	

メソッドの詳細

convertPosToCol():

public int convertPosToCol(int pos)

このメソッドは、表示スペースの指定された位置に関連する列番号を戻します。

<message title="'Example of convertPosToCol()'" value="'Cursor is at column '+ \$HMLPSUtil.convertPosToCol(\$HMLPSUtil.getCursorPos()\$)\$" />

図 59. convertPosToCol() の例

convertPosToRow():

public int convertPosToRow(int pos)

このメソッドは、表示スペースの指定された位置に関連する行番号を戻します。

<message title="'Example of convertPosToRow()'"
 value="'Cursor is at row '+
 \$HMLPSUtil.convertPosToRow(\$HMLPSUtil.getCursorPos()\$)\$" />

図 60. convertPosToRow() の例

enableRoundTrip():

public void enableRoundTrip(boolean flag)

このメソッドは、双方向言語専用 (アラビア語およびヘブライ語) です。A、B、およ び C が双方向文字だと仮定します。通常、ストリングに一続きの双方向文字が含ま れ、次に、一続きの数表示が続き (例えば、ABC 123)、ストリング全体が格納される 場合、Host On-Demand クライアントは、双方向文字と数表示の位置を交換しま す。例えば、通常、ストリング ABC 123 を表示スペースから読み出してストリング を変数に格納し、次に、変数の値を表示スペースに書き込む場合、Host On-Demand は表示スペースに 123 ABC と書き込みます。

強制的に反転する機能をオフにするには、enableRoundTrip() の値を true にして 呼び出します。強制的に反転する機能をオンにするには、enableRoundTrip() の値 を false にして呼び出します。

<perform value="\$HMLPSUtil.enableRoundTrip(true)\$" />

図 61. enableRoundTrip() の例

getCursorCol():

public int getCursorCol()

このメソッドは、表示スペースのテキスト・カーソルの列の位置を戻します。

```
<input value="$HMLSessionUtil.getHost()$"
row="$HMLPSUtil.getCursorRow()$"
col="$HMLPSUtil.getCursorCol()$+2"
movecursor="true" xlatehostkeys="true"
encrypted="false" />
```

図 62. getCursorCol()の例

getCursorPos():

public int getCursorPos()

このメソッドは、表示スペースのテキスト・カーソルの位置を戻します。

図 63. getCursorPos()の例

getCursorRow():

public int getCursorRow()

このメソッドは、表示スペースのテキスト・カーソルの行の位置を戻します。

```
<input value="$HMLSessionUtil.getHost()$"
row="$HMLPSUtil.getCursorRow()$"
col="$HMLPSUtil.getCursorCol()$+2"
movecursor="true" xlatehostkeys="true"
encrypted="false" />
```

図 64. getCursorRow()の例

getSize():

public int getSize()

このメソッドは、表示スペースのサイズ、つまり表示スペースの文字位置の数を戻します。例えば、セッション・ウィンドウが 25 行と 80 列の場合、表示スペースのサイズは 24 * 80 = 1920 になります。

```
<message title="'Example of getSize()'"
value="'Size of PS is '+$HMLPSUtil.getSize()$" />
```

図 65. getSize() の例

getSizeCols():

public int getSizeCols()

このメソッドは、表示スペースの列数を戻します。表示スペースの列数は、セッション・ウィンドウの列数と同じになります。例えば、セッション・ウィンドウが 25 行と 80 列の場合、表示スペースの列数は 80 になります。

```
<message title="'Example of getSizeRows and getSizeCols()'"
    value="'The PS has '+$HMLPSUtil.getSizeRows()$+
    ' rows and '+
    $HMLPSUtil.getSizeCols()$+' columns'" />
```

図 66. getSizeCols() の例

getSizeRows():

public int getSizeRows()

このメソッドは、表示スペースの行数を戻します。表示スペースの行数は、セッション・ウィンドウの行数よりも 1 つ少なくなります (これは、セッション・ウィンドウの最後の行にはオペレーター情報域があるので、表示スペースに含まれないためです)。例えば、セッション・ウィンドウが 25 行と 80 列の場合、表示スペースの行数は 24 になります。

```
<message title="'Example of getSizeRows and getSizeCols()'"
    value="'The PS has '+$HMLPSUtil.getSizeRows()$+
    ' rows and '+
    $HMLPSUtil.getSizeCols()$+' columns'" />
```

図 67. getSizeRows() の例

getString():

public String getString(int pos, int len)

このメソッドは、表示スペースの指定された位置から開始される、指定された文字 数のテキスト・ストリングを戻します。

```
<message title="'Text of row 18:'"
value="'Text:'+$HMLPSUtil.getString(
$HMLPSUtil.getSizeCols()$*17+1,
$HMLPSUtil.getSizeCols()$)$" />
```

図 68. getString() の例

searchString():

public int searchString(int pos, int len)

このメソッドは、指定されたストリングの表示スペースの位置を戻します。このメ ソッドは、ストリングが表示スペース内に検出されないと0を戻します。

```
<varupdate name="$int$" value="$HMLPSUtil.searchString('IBM')$" />
<message title="'Searching for IBM'"
    value="'IBM is found at position '+$int$+
    ' that is row='+
    $HMLPSUtil.convertPosToRow($int$)$+
    ', column='+
    $HMLPSUtil.convertPosToCol($int$)$" />
```

図 69. searchString() の例

\$HMLSessionUtil\$

\$HMLSessionUtil\$ を使用して呼び出されたメソッドは、セッションに関連した値 を戻します。表 29 には、これらのメソッドの要約が記載されています。

表 29. \$HMLSessionUtil()\$ のメソッドの要約

メソッドの要約	メソッドの要約 : \$HMLSessionUtil\$		
ストリング (String)	getHost() セッション構成の「 宛先アドレス (Destination Address) 」フィー ルドで指定されたテキスト・ストリングを戻します。		
ストリング (String)	getLabel() セッション構成の「 セッション名 (Session Name) 」フィールドで指 定されたストリングを戻します。		
ストリング (String)	getName() ホストがセッションに割り当てたセッション・インスタンス ID を 戻します。		

メソッドの詳細

getHost():

public String getHost()

このメソッドは、セッション構成の「接続」セクションの「**宛先アドレス**」フィー ルドに入力したホスト名またはホスト・アドレスを戻します (myhost.myloc.mycompany.com または 9.27.63.45 など)。

<message title="'Host name or address'" value="\$HMLSessionUtil.getHost()\$" />

図 70. getHost() の例

getLabel():

public String getLabel()

このメソッドは、セッション構成の「接続」セクションの「**セッション名 (Session** Name)」フィールドに入力したセッション名を戻します (3270 Display または 5250 Display など)。

<message title="'Session name'" value="\$HMLSessionUtil.getLabel()\$" />

図 71. getLabel() の例

getName():

public String getName()

このメソッドは、ホストがセッションに割り当てた識別名 (A、B、または C など) を戻します。セッションを開始するとき、ホストは名前をセッションに割り当て て、開始している可能性のある同じセッションの他のインスタンスと区別します。

<message title="'Session instance identifier'" value="\$HMLSessionUtil.getName()\$" />

図 72. getName() の例

\$HMLSQLUtil\$

\$HMLSQLUtil\$ で呼び出されたメソッドは、最新の SQLQuery アクションの結果 に関する情報を戻します。表 30 には、これらのメソッドの要約が記載されていま す。

表 30. \$HMLSessionUtil()\$ のメソッドの要約

メソッドの要約 : \$HMLSessionUtil\$		
int	getColumnSize() データの列数を戻します。	
ストリング (String)	getDataByIndex() 指定された行のインデックスと列のインデックスに配置された項目 を戻します。	
ストリング (String)	getDataByName() 指定された行のインデックスと列名 (フィールド名) に配置された 項目を戻します。	
int	getRowSize() データの行数を戻します。	

保管データの形式

SQLQuery アクションの結果は、戻されたデータのブロックのサイズよりもそれぞ れ 1 ずつ広い列および高い行から構成される、2 次元配列として保管されます。行 0 は列名 (データベースからのフィールド名) を保管するのに使用されて、列 0 は ゼロ・ベースのインデックスを保管するのに使用されます (下の表 31 を参照)。行 0、列 0 の項目には空ストリングが含まれます。配列の残りには実際のデータが含 まれます。すべての値はストリングです。

表 31 には、クエリーの結果である 3 x 5 のデータのブロックが、4 x 6 の配列に 保管される例が記載されています。

(空ストリ ング)	TOPICID	EXMPLID	DESCRIPT
0	4	18	Create a toolbar with custom buttons.
1	9	54	Attach tables at startup.
2	11	74	Edit Products.
3	11	75	Enter or Edit Products
4	11	76	Find Customers

表 31. 結果を含んだ 2 次元配列の例

上の表では、行 0、列 0 の項目には空ストリングが含まれています。行 0 以降に は、データベースからのフィールド名 (TOPICID, EXMPLID, DESCRIPT) が含まれ ています。列 0 以降には、行のインデックス番号 (0, 1, 2, 3, 4) が含まれていま す。配列 0 以降には、実際のデータが含まれています。すべての値はストリングで す。

メソッドの詳細

getColumnSize():

public int getColumnSize()

このメソッドは、追加された列 0 を含む、配列内のデータの実際の列数を戻しま す。例えば、表 31 の配列では、このメソッドは 4 を戻します。

<message title="'Column size'" value="\$HMLSessionUtil.getColumnSize()\$" />

図 73. getColumnSize() の例

getDataByIndex():

public int getDataByIndex(int row, int column)

このメソッドは、指定された行と列のインデックスの項目を戻します。表 31 のデ ータに対して戻された値のリストは以下のとおりです。

- getDataByIndex(0,0) は空ストリングを戻す。
- getDataByIndex(0,1) はストリング「TOPICID」を戻す。
- getDataByIndex(0,2) はストリング「EXMPLID」を戻す。
- getDataByIndex(1,1) はストリング「4」を戻す。

- getDataByIndex(2,2) はストリング「54」を戻す。
- getDataByIndex(3,3) はストリング「Edit Products」を戻す。

<message title="'Row 3, Column 3'" value="\$HMLSessionUtil.getDataByIndex(3,3)\$" />

図 74. getDataByIndex() の例

getDataByName():

public int getDataByName(int row, String fieldName)

このメソッドは、指定された行とフィールド名の項目を戻します。 184 ページの 表 31 のデータに対して戻された値のリストは以下のとおりです。

- getDataByIndex(1, TOPICID) はストリング「4」を戻す。
- getDataByIndex(2, EXMPLID) はストリング「54」を戻す。
- getDataByIndex(3, DESCRIPT) はストリング「Edit Products」を戻す。

```
<message title="'Row 3, Field DESCRIPT'"
value="$HMLSessionUtil.getDataByName(3,'DESCRIPT')$" />
```

図 75. getDataByName() の例

getRowSize():

public int getRowSize()

このメソッドは、追加された行 0 を含む、配列内のデータの実際の行数を戻しま す。例えば、 184 ページの表 31 の配列では、このメソッドは 6 を戻します。

<message title="'Column size'" value="\$HMLSessionUtil.getRowSize()\$" />

図 76. getRowSize() の例

\$HMLSystemUtil\$

\$HMLSystemUtil\$ で起動されるメソッドで、アプレット、Java プロパティー、および OS 環境に関する情報を得ることができます。

表 32 には、これらのメソッドの要約が記載されています。

表 32. \$HMLSystemUtil()\$ のメソッドの要約

	メソッ	ドの要約:	\$HMLS	vstemUtil\$
--	-----	-------	--------	-------------

ストリング	getHTMLParameter(String name)
(String)	HTML パラメーター名を受け入れ、その値を戻します。

表 32. \$HMLSystemUtil()\$ のメソッドの要約 (続き)

メソッドの要約: \$HMLSystemUtil\$				
ストリング (String)	getenv(String name) オペレーティング・システムの環境変数名を受け入れ、その値を戻 します。			
ストリング (String)	getSystemProperty(String name) Java システム・プロパティー名を受け入れ、その値を戻します。			

メソッドの詳細

getHTMLParameter():

public String getHTMLParameter(String name)

このメソッドは、HTML パラメーター名を受け入れ、その値を戻します。

存在しないパラメーター名を受け取ると、メソッドは空のストリングを戻します。

\$HMLSystemUtil.getHTMLParameter('some_param_name')\$

図 77. getHTMLParameter() の例

getenv():

public String getenv(String name)

このメソッドは、オペレーティング・システムの環境変数名を受け入れ、その値を 戻します。

存在しない変数名を受け取ると、メソッドは空のストリングを戻します。

\$HMLSystemUtil.getenv('Temp')\$

図 78. getenv() の例

制限:

- 1. Java 1.5.0 より前のバージョンを使用している場合、変数名と同じ値を持つ環 境変数が定義されていると、その変数は存在せず戻ってきたものと見なされて空 のストリングが戻されます。
- 2. Java 1.5.0 より前のバージョンを使用している場合、Windows 以外のプラット フォームでは、常に空のストリングが戻されます。

getSystemProperty():

public String getSystemProperty(String name)

このメソッドは、Java システム・プロパティー名を受け入れ、その値を戻します。

存在しない変数名を受け取ると、メソッドは空のストリングを戻します。

\$HMLSystemUtil.getSystemProperty('java.home')\$

図 79. getSystemProperty() の例

一般的な制限: 大/小文字の区別があるオペレーティング・システムを使用してい る場合、これらのメソッドのパラメーターとして指定する名前も大/小文字の区別が あります。

FormatNumberToString() および FormatStringToNumber()

\$FormatNumberToString()\$ は推奨されておらず、代わりに
\$HMLFormatUtil.numberToString()\$ の使用をお勧めします。両者の入力パラメータ
-と戻り値の型は同じです (175 ページの『numberToString()』を参照)。

\$FormatStringToNumber()\$ は推奨されておらず、代わりに
\$HMLFormatUtil.stringToNumber()\$ の使用をお勧めします。両者の入力パラメータ
-と戻り値の型は同じです (176 ページの『stringToNumber()』を参照)。

第12章 その他のセキュリティー・オプション

この章では、その他のセキュリティー・オプションについて説明します。

パスワードの記録

「パスワードの記録」オプションはデフォルトで使用可能です。「パスワードの記録」は、次のいずれかの方法で使用不可にすることができます。

- デプロイメント・ウィザードの「機能の使用不可 (Disable Functions)」ウィンド ウの「マクロ (Macro)」セクションで、「パスワードの記録」オプションに対応 する「使用不可」をクリックする。
- 管理ユーティリティーの「機能の使用不可 (Disable Functions)」ウィンドウの 「マクロ (Macro)」セクションで、「パスワードの記録」オプションに対応する 「使用不可」をクリックする。

「パスワードの記録」オプションは、マクロ記録中にエンド・ユーザーが 3270 または 5250 非表示入力フィールドに入力した入力キー・シーケンスに関する Host On-Demand の記録方法を制御します。非表示入力フィールドは、入力時に文字が表示されない 3270 または 5250 フィールドです。通常、非表示入力フィールドはパスワードなどの機密情報を入力するときに使用されます。

「パスワードの記録」オプションについて、次の点に注意してください。

- 3270 表示セッションおよび 5250 表示セッションにのみ影響する。
- マクロ記録中のみ有効 (ただし、一部の効果はマクロ再生中も有効)。
- マクロ記録中に 3270 および 5250 非表示フィールドに入力された入力キー・シ ーケンスにのみ影響する。

「パスワードの記録」が使用可能な場合

「パスワードの記録」が使用可能な場合 (デフォルト設定) に、エンド・ユーザーが マクロ記録中に 3270 または 5250 非表示入力フィールドに文字列を入力すると、 Host On-Demand は入力キー・シーケンスを入力アクションとして記録し、入力ア クションの「パスワード」オプションを使用して、入力アクション内の入力キー・ シーケンスを暗号化します。

例えば、「パスワードの記録」が使用可能な場合、マクロの記録中に、3270 または 5250 非表示入力フィールドにログオン・パスワード ("MyPassword" など) を入力 するとします。 Host On-Demand は入力アクションを作成し、ログオン・パスワ ードを暗号化して入力アクションに格納し、入力アクションを記録中のマクロ・ス クリプトに書き込みます。

入力アクション内の入力キー・シーケンスを暗号化すると、無許可のユーザーが記録されたマクロ・スクリプトを編集または表示して、偶然に入力キー・シーケンスを検出することができなくなるため、ある程度のセキュリティーが実現します。 Host On-Demand は、マクロ再生中に入力アクションを処理する場合を除き、入力アクション内の入力キー・シーケンスの暗号化を解除しません。ただし、利口な無 許可ユーザーは、マクロ・スクリプトにアクセス可能であれば、マクロ再生中に入 カキー・シーケンスを通常の表示可能フィールドに転送するように入力アクション を編集して、入力キー・シーケンスを検出できます (99 ページの『パスワード』 を参照)。

入力キー・シーケンスを入力アクションに記録することの利点は、ユーザーが操作 しなくても、マクロ再生中にマクロが自動実行されることです。ただし、必須のキ ー・シーケンスが変更された場合 (例えば、パスワードの期限が切れて、異なるパ スワードを使用する必要がある場合) は、新しい入力キー・シーケンスで入力アク ションを更新する必要があります。

「パスワードの記録」が使用不可な場合

「パスワードの記録」が使用不可な場合に、エンド・ユーザーがマクロ記録中に 3270 または 5250 非表示入力フィールドに入力キー・シーケンスを入力すると、 Host On-Demand は入力キー・シーケンスを廃棄して、記録されたマクロ・スクリ プト内に入力アクションでなくプロンプト・アクションを作成します。プロンプ ト・アクションには入力キー・シーケンスの長さ、および入力キー・シーケンスの 転送先となる画面上の行および列の位置に関する情報が格納されますが、入力キ ー・シーケンス自体は格納されません。したがって、無許可ユーザーは、マクロ・ スクリプトを表示、編集、または実行して入力キー・シーケンスを検出することが できません。

マクロ再生中に、Host On-Demand がプロンプト・アクションを処理する場合、入 カフィールドおよびエンド・ユーザーに入力を要求するメッセージが表示されたプ ロンプト・ウィンドウがポップアップします (デフォルト・メッセージは 「Password」ですが、マクロ記録が完了したら、プロンプト・アクションを編集し て別のメッセージを指定できます)。エンド・ユーザーがプロンプト・ウィンドウの 入力フィールドに入力キー・シーケンスを入力して、「OK」をクリックすると、 Host On-Demand はプロンプト・ウィンドウを閉じて、指定された行および列に配 置されたセッション・ウィンドウ内の 3270 または 5250 非表示フィールドに同じ 入力キー・シーケンスを入力します。

したがって、「パスワードの記録」を使用不可にすると、入力キー・シーケンスが マクロ・スクリプトに保管されない (暗号化形式で保管されることもない) ため、使 用可能にした場合よりもセキュリティーが高まります。その代わりに、Host On-Demand がプロンプト・アクションを処理する場合、エンド・ユーザーはマク ロ再生中に入力キー・シーケンスを入力する必要があります。

Host On-Demand がマクロ記録中にプロンプト・アクションを作成すると、プロン プト・アクションの「パスワード応答」オプションが有効になります。このオプシ ョンを有効にすると、マクロ再生中にユーザーが入力キー・シーケンスを入力した 場合、プロンプト・ウィンドウの入力フィールドに表示される文字列にはアスタリ スク (*) が使用されます (116 ページの『パスワード応答』を参照)。

第13章 その他の拡張機能

この章では、その他の拡張機能について説明します。

複数のセッションとの対話

Host On-Demand バージョン 9.0 では、1 つのマクロで、そのマクロを起動した ディスプレイ・エミュレーション・セッション以外のディスプレイ・エミュレーシ ョン・セッションと対話できます。

この機能により、異なるホスト・アプリケーション間のデータの受け渡し、および 複数のホスト・アプリケーションから単一のワークステーション・アプリケーショ ンへのデータの受け渡しが、簡単に素早く行えるようになります。 例えば、次のと おりです。

- 1 つのマクロで、3270 ディスプレイ・セッション内のアプリケーションからデ ータを収集し、5250 ディスプレイ・セッション内のアプリケーションとの対話 でそのデータを使用して、次に、3270 ディスプレイ・セッションでの作業に戻 ることができる。
- 1 つのマクロで、VT ディスプレイ・セッションおよび 3270 ディスプレイ・セッションからデータを収集し、次に、ワークステーション・アプリケーションを 起動して、両方のデータのセットを処理することができる。

この機能が使用可能になる以前でも、同様のタスクを行うことはできましたが、複 雑でかつ自動化も限られていました。それぞれのホスト・セッションごとに個別の マクロが必要であり、各マクロを手動で起動する必要がありました。また、あるマ クロの作業結果が完了してから、その作業結果に依存するマクロを起動するなど、 マクロの同期を手動で行う必要がありました。

基本

複数のセッションと対話するマクロは、一般のマクロが保管されているどの場所に も保管することができます。

複数のセッションと対話するマクロは、一般のマクロと同様に、単一セッションに よって起動されます。ただし、そのマクロと対話するすべてのセッションは、マク ロが起動される前に、アクティブである (接続されている) 必要があります (195 ページの『ランタイム要件』 を参照)。

マクロが対話するセッションはさまざまで、異なるタイプ (3270 ディスプレイ、 5250 ディスプレイ、VT ディスプレイ、または CICS ゲートウェイ) の場合、また はすべて同じタイプの場合があります。また、別々のホストに接続する場合も、同 じホストに接続する場合もあります。

単一のマクロが対話できるアクティブ・セッションの数に制限はありませんが、以 下の条件があります。

 すべてのセッションが、単一の Host On-Demand クライアントのデスクトップ から起動されるか、またはすべてのセッションが同一の JVM から起動される。 Host On-Demand クライアントの起動に使用された Java 仮想マシンのメモリ ーが使い果たされていない。

実装

この機能をサポートするマクロ・ディスクリプターおよびマクロ・アクションに、 ホスト ID フィールドが追加されました。ホスト ID フィールドの値により、ディ スクリプターやアクションが適用されるアクティブ・セッションが識別されます (196 ページの『ホスト ID の指定』を参照)。

マクロを再生しているときに、マクロ・ランタイムは、ディスクリプターの評価や アクションの実行の準備時に、ディスクリプターやアクションのホスト ID フィー ルドを確認します。ホスト ID フィールドが欠落している場合または空白の場合、 マクロ・ランタイムはディスクリプターやアクションをそのマクロを起動したセッ ションである、オリジナル・セッション (例えば 3270 ディスプレイ・セッション) に適用します。ただし、ホスト ID フィールドが指定されていて非ブランクである 場合、マクロ・ランタイムは次のように動作します。

- 1. ホスト ID フィールドが参照するアクティブ・セッション (例えば、別のホスト に接続された 2 番目の 3270 ディスプレイ・セッション) を探し出す。
- ディスクリプターやアクションを、そのアクティブ・セッションのセッション・ ウィンドウに適用する。

この機能は以下のディスクリプターでサポートされています。

- OIA が禁止解除になるのを待つ (Wait for OIA to Become Uninhibited)
- フィールド数 (Number of Fields)
- 入力フィールド数 (Number of Input Fields)
- ストリング (String)
- カーソル (Cursor)
- 属性 (Attribute)

上記のディスクリプターを使用することにより、マクロを起動したオリジナル・セ ッション以外のアクティブ・ディスプレイ・エミュレーション・セッションを参照 することができます。残りのディスクリプター (条件、カスタム) および可変アップ デート・アクションは特定のセッション・ウィンドウを参照しないため、この機能 はサポートされていません。上記リスト中の最初の 3 つのディスクリプターでは、 マクロ・エディター中に表示 されるディスクリプターの数に制限があります (200 ページの『同一のマクロ画面での異なるセッションの参照』を参照)。

以下のマクロ・アクションでは、この機能がサポートされています。

- ボックス選択
- 通信待機
- 抽出
- 入力
- マウス・クリック
- 印刷抽出
- プロンプト
- 転送

上記のアクションを、マクロを起動したオリジナル・セッション以外のアクティ ブ・ディスプレイ・エミュレーション・セッションに適用できます。残りのアクシ ョン (条件、ファイル・アップロードなど) は特定のセッション・ウィンドウ上で作 動しないので、この機能はサポートされていません。

マクロ画面作成の 2 つの手法

マクロを起動したオリジナル・セッション以外のセッションを参照するマクロ画面 を作成する場合、少なくとも 2 つの手法を使用することができます。

まず、1 つのアクティブ・セッションのみを参照するマクロ画面を、それぞれ作成 する手法があります。つまり、ある 1 つのマクロ画面内のすべてのディスクリプタ ー、およびすべてのアクションによって、同一のアクティブ・セッションが参照さ れます。この手法により、優れた柔軟性が得られ、マクロ画面の作成、デバッグ、 および、保守が簡単になります。

例えば、作成したマクロのうち、最初の 10 個のマクロ画面はオリジナル・セッションと、次の 8 個のマクロ画面は 2 番目のセッションと、最後の 4 個のマクロ画面は再度オリジナル・セッションと対話することができます。

この第 1 の手法は、複数のアクティブ・セッションと対話するマクロを記録すると きの、記録済みの素材を格納する場合に、マクロ・オブジェクトによって使用され ます。あるアクティブ・セッションから他のセッションに切り替えられるたびに、 マクロ・オブジェクトは現行のマクロ画面の完了と格納を行い、新規のマクロ画面 を作成することにより、新しく選択されたセッションの最初のアプリケーション画 面用のディスクリプターとアクションを保持します (197 ページの『記録済みマク ロ画面は単一のセッション・ウィンドウを参照』を参照)。

このように、複数のセッションを参照するマクロを記録する場合、マクロ・オブジ ェクトは第1の手法を使用してマクロを作成します。

次に、マクロ画面で、複数のアクティブ・セッションを参照する手法があります。 ディスクリプターが複数のアクティブ・セッションを参照するか、アクションが複 数のアクティブ・セッションを参照するか、または両者が行います。ディスクリプ ターが複数のアクティブ・セッションを参照する場合、マクロ・ランタイムは、指 定されたセッションのセッション・ウィンドウを参照する各ディスクリプターを評 価します。アクションが複数のアクティブ・セッションを参照する場合、マクロ・ ランタイムは、指定されたセッションのセッション・ウィンドウに対する各アクシ ョンを実行します。

第2の手法は、1つのマクロ画面内に複数のセッションへの参照を追加するため、 マクロ・エディターかコード・エディターのいずれかによるマクロの編集が必要で す(200ページの『同一のマクロ画面での異なるセッションの参照』を参照)。

第 2 の手法は第 1 の手法よりも柔軟であり、状況に応じて必要となる場合があり ます。ただし、この第 2 の手法は、一般的にマクロ画面の作成やデバッグ、および 保守を行う上で多少複雑になります。

ランタイムの処理

このセクションでは、マクロ再生時の複数セッションにおけるディスクリプターお よびアクションの処理の詳細について説明し、理解を深めます。

ディスクリプター

マクロ・ランタイムが複数のセッションを参照するマクロを再生する場合、マク ロ・ランタイムは、通常のマクロの場合とまったく同様に、ディスクリプターを処 理します。 ただし、ディスクリプターがマクロを起動したオリジナル・セッション 以外のアクティブ・セッションに適用される場合、マクロ・ランタイムは、そのア クティブ・セッションのセッション・ウィンドウに対して、ディスクリプターが真 であるか偽であるかを評価します。これには 2 つの主なケースがあり、次の 2 つ のサブセクションで説明します。

すべてのディスクリプターが同じアクティブ・セッションに適用される: マクロが 3270 ディスプレイ・セッションから起動され、マクロ・ランタイムが、その 3270 ディスプレイ・セッションを参照するマクロ画面の処理を終了して、次に処理する マクロ画面の認識を試行中であると仮定します。また、マクロ・ランタイムは Screen26 という名前のマクロ画面を候補として確認中であり、その Screen26 が、 それぞれ同一のアクティブな 5250 ディスプレイ・セッションに適用される以下の ディスクリプターを含んでいると仮定します。

- 「フィールド数 (Number of Fields)」ディスクリプター
- 「入力フィールド数 (Number of Input Fields)」ディスクリプター
- 「OIA が禁止解除になるのを待つ (Wait for OIA to Become Uninhibited)」ディスクリプター

上記 3 つのディスクリプターはすべて必要なディスクリプターであり、逆ディスク リプターはないと仮定します。また、各ディスクリプターが 5250 ディスプレイ・ セッションのセッション・ウィンドウに対し、真であると評価された場合、マク ロ・ランタイムは、 Screen26 が次に処理するマクロ画面であると決定します。例 えば、ディスクリプターの値が 4、53、および NOTINHIBITED であると仮定しま す。5250 ディスプレイ・セッションのセッション・ウィンドウが 4 つの入力フィ ールド、およびすべてのタイプの 53 のフィールドを含み、入力禁止標識がクリア されている場合、マクロ・ランタイムは Screen26 が次に処理するマクロ画面であ ると決定します。

上記の処理は、ディスクリプターがマクロを起動したセッションのセッション・ウ ィンドウ以外のセッション・ウィンドウに適用されている点を除いて、通常のマク ロ画面の処理とまったく同様です。

ディスクリプターが異なるアクティブ・セッションに適用される: 再度、マクロが 3270 ディスプレイ・セッションから起動され、マクロ・ランタイムが、その 3270 ディスプレイ・セッションを参照するマクロ画面の処理を終了して、次に処理する マクロ画面の認識を試行中であると仮定します。また、マクロ・ランタイムは Screen32 という名前のマクロ画面を候補として確認中であり、その Screen32 が、 以下の 6 つのディスクリプターを含んでいると仮定します。

 すべて、オリジナルの 3270 ディスプレイ・セッションに適用される「フィール ド数 (Number of Fields)」ディスクリプター、「入力フィールド数 (Number of Input Fields)」ディスクリプター、および「OIA が禁止解除になるのを待つ (Wait for OIA to Become Uninhibited)」ディスクリプター

 すべて、5250 ディスプレイ・セッションに適用される「フィールド数 (Number of Fields)」ディスクリプター、「入力フィールド数 (Number of Input Fields)」ディスクリプター、および「OIA が禁止解除になるのを待つ (Wait for OIA to Become Uninhibited)」ディスクリプター

上記のディスクリプターはすべて必要なディスクリプターであり、逆ディスクリプ ターはないと仮定します。さらに、以下の条件の場合、マクロ・ランタイムは、 Screen32 が次に処理するマクロ画面であると決定します。

- 最初の 3 つのディスクリプターが、3270 ディスプレイ・セッション・ウィンド ウに対して真であると評価される。さらに、
- 後の 3 つのディスクリプターが、5250 ディスプレイ・セッション・ウィンドウ に対して真であると評価される。

この例において、個々のディスクリプターを評価する処理は、マクロ・ランタイム が2つの異なるアクティブ・セッション・ウィンドウに対しディスクリプターを評 価する点を除いて、通常のマクロとまったく同様です。

特定のセッション・ウィンドウに対し、各ディスクリプターが真であるか偽である かを評価した後、マクロ・ランタイムは、通常の方法で、個々のディスクリプター のブールの結果を結合します (58 ページの『ディスクリプターの評価』を参照)。

アクション

マクロ・ランタイムが複数のセッションを参照するマクロを再生する場合、マク ロ・ランタイムは、通常のマクロの場合とまったく同様に、アクションを処理しま す。 ただし、アクションに、マクロを起動したオリジナル・セッション以外のアク ティブ・セッションを参照するホスト ID が含まれる場合、マクロ・ランタイム は、そのアクティブ・セッションのセッション・ウィンドウに対するアクションを 実行します。

ディスクリプターと同様に単一のマクロ画面では、同じセッションに適用されるす べてのアクション、または複数の異なるセッションに適用されるアクションを含ま せることができます。 例えば、マクロ画面に、3270 ディスプレイ・セッションな どの、マクロを起動したオリジナル・セッションに対する 1 番目の入力アクショ ン、および、 5250 ディスプレイ・セッションなど、別のセッションに対する 2 番 目の入力アクションが含まれる場合、マクロ・ランタイムは以下のようになりま す。

- 1 番目の入力アクションからの入力キー・シーケンスを、オリジナル・セッション・ウィンドウ (3270 ディスプレイ・セッションのセッション・ウィンドウ)の指定された行と列の位置に入力します。
- 2 番目の入力アクションからの入力キー・シーケンスを、2 番目のセッション・ ウィンドウ (5250 ディスプレイ・セッションのセッション・ウィンドウ)の指 定された行と列の位置に入力します。

ランタイム要件

エンド・ユーザーが複数のセッションと対話するマクロを実行する場合、マクロ・ ランタイムには次の 2 つの要件があります。

- マクロが対話する各セッションは、エンド・ユーザーがマクロを起動する前に開 始済みで、接続されている必要がある。
- マクロが対話する各セッションは、マクロが期待する最初のアプリケーション画面に配置される必要がある。

最初の要件のため、エンド・ユーザーによりマクロが起動される場合には、常にマ クロ・ランタイムは、マクロ実行の準備の一部として、マクロ・スクリプト全体の ホスト ID フィールドを走査し、マクロがアクセスするセッションのリストを収集 します。次に、マクロ・ランタイムはリスト中の各セッションが開始済みで、接続 されていることを検査します。セッションの 1 つが開始済みではなく、接続されて いない場合には、マクロ・ランタイムはエラー・メッセージを表示し、マクロ再生 を終了します。

2 番目の要件は、エンド・ユーザーが、マクロを起動したセッションを、そのマク ロの最初のマクロ画面が期待するアプリケーション画面に配置するという、通常の 要件の拡張です。通常のマクロの最初のマクロ画面は、特定のアプリケーション画 面に配置されたセッション・ウィンドウを検出することを想定しています。同様 に、オリジナル・セッション以外のセッションと対話する最初のマクロ画面も、特 定のアプリケーション画面に配置されたセッションのセッション・ウィンドウを検 出することを想定しています。

ホスト ID の指定

マクロ・エディターでは、ホスト ID 入力フィールドは、マクロを起動したオリジ ナル・セッション以外のセッションを参照できる、各ディスクリプターまたはアク ションの編集ウィンドウに表示されます (192 ページの『実装』のディスクリプタ ーおよびアクションのリストを参照)。コード・エディターでは、対応する XML エ レメントの hostid 属性が有効です。

ホスト ID は次のもので構成されています。

- 1. セッション ID (A など)。以下が後に続く。
- 2. コロン (:)。以下が後に続く。
- 3. セッション名 (3270 Display など)。

例: B:3270 Display、C:5250 Display。

エンド・ユーザーがセッションを起動したときに Host On-Demand クライアント が自動的にセッション ID の割り当てを行うのではなく、マクロがオリジナル・セ ッション以外を参照する各セッションのセッション・プロパティー中に、特定のセ ッション ID (A から Z までの 1 文字) をセッションとして指定することをお勧め します。セッションに特定のセッション ID を指定するには、次のように行いま す。

- 1. セッションのプロパティー・ウィンドウを開きます。
- 2. 左ペインの「開始オプション (Start Options)」をクリックします。
- 右ペインの「セッション ID リスト・ボックス (Session ID listbox)」内で、値 を「自動 (Automatic)」から使用する値 (A から Z) に変更します。

上記の代わりに Host On-Demand クライアントがセッション ID を自動的に割り 当てるようにした場合、セッション ID は、マクロ・スクリプトで使用しているセ ッション ID (例えば B) とは異なる値 (例えば C) になります。また、その場合に は、ホスト ID の値も異なる値 (B:3270 Display ではなく C:3270 Display) になり ます。その結果、エンド・ユーザーがマクロを起動すると、マクロ・ランタイム は、マクロにアクティブでなく接続されていないセッションへの参照 (B:3270 Display) が含まれるために、エラーによりマクロ再生を終了させる場合がありま す。

ホスト ID のセッション名コンポーネントを変更する (例えば、3270 Display を SessionApp1 に変更する) には、次のように行います。

- 1. セッションのプロパティー・ウィンドウを開きます。
- 2. 左ペインの「接続」をクリックします。
- 3. 右ペインの「セッション名 (Session Name)」フィールドに、使用するセッショ ン名を入力します。

異なるセッションでの自動編集機能の使用

マクロ・エディターにおいて、ディスクリプターの編集ウィンドウのホスト ID フ ィールドに、参照セッションを入力した後、ホスト ID により参照されたセッショ ンがアクティブであり、かつディスクリプターに「現行 (Current)」ボタンがある場 合、「現行 (Current)」ボタンを使用し、ホスト ID が参照するセッションのセッシ ョン・ウィンドウから自動的にデータを収集できます。

ただし、同じ方法で、マーキング長方形を使用することはできません。つまり、マ ーキング・ウィンドウを使用して、マクロ・エディターを起動したセッションに属 するセッション・ウィンドウ以外の、あらゆるセッション・ウィンドウからの編集 情報を収集することはできません。

複数のセッションと対話するマクロの記録

このセクションでは、複数のセッションと対話するマクロを記録する場合の問題に ついて説明します。次の各サブセクションでは、それぞれ 1 つの問題を取り上げて います。

マクロ・スクリプトの格納

通常のマクロと同様に、マクロ・オブジェクトは記録済みマクロのマクロ・スクリ プトを選択された場所に作成します。

記録処理の実行中に別のアクティブ・セッションに切り替えた場合、マクロ・オブ ジェクトは継続して、最初に選択された同一の場所へ記録情報を格納します。

記録済みマクロ画面は単一のセッション・ウィンドウを参照

記録処理の実行中に、あるアクティブ・セッションから別のアクティブ・セッションに切り替える場合、マクロ・オブジェクトは次のようになります。

- 1. 現行のマクロ画面を終了し、格納する。
- 2. デフォルトの記録済みディスクリプター (OIA が禁止解除になるのを待つ、フィールド数、および入力フィールド数) で構成された新規のマクロ画面を作成する。
- 新たに選択されたセッションからのアクションが、新規マクロ画面に追加されるのを待つ。

したがって、マクロ画面の作成で前に説明した 2 つの手法のうちの第 1 の手法、 つまり、同一のアクティブ・セッションを参照する 1 つのマクロ画面の中ですべて のディスクリプターおよびアクションを作成する手法が、マクロ・オブジェクトで 使用されます (193 ページの『マクロ画面作成の 2 つの手法』を参照)。

開始、停止、および、セッションの切り替え

通常どおり、マクロ・マネージャー・ツールバーの「マクロを記録 (Record macro)」アイコンをクリックし、記録を開始します。記録は、記録の完了後にマクロを起動する予定のセッションと同一のセッションで開始する必要があります。

別のセッションに切り替えるには、記録するセッションのマクロ・マネージャー・ ツールバー上の「記録を追加 (Append recording)」アイコンをクリックします。最 初のセッションに切り替えるには、最初のセッションのマクロ・マネージャー・ツ ールバー上の「記録を追加 (Append recording)」アイコンをクリックします。

任意の時点で記録を停止するには、最初に記録を開始した同一のセッションのマク ロ・マネージャー・ツールバー上の「記録を停止 (Stop recording)」アイコンをク リックします。例えば、3270 ディスプレイ・セッションで記録を開始し、さらに、 2 つの別のアクティブな 3270 ディスプレイ・セッションで記録を継続した場合、 記録を停止するには、最初の 3270 ディスプレイ・セッション上の「記録を停止 (Stop recording)」アイコンをクリックする必要があります。

開始、セッションの切り替え、および停止についての上記の要約は次の 表 33 のと おりです。

機能:	クリックするアイコン:	アイコンが配置されている
		セッション:
記録を開始	「記録を開始 (Start	完成したマクロを実行する予
	recording)	定のセッション。
すべてのアクティブ・セッシ	「記録を追加 (Append	切り替え先のセッション。
ョン (最初のセッションを含	recording)]	
む) への切り替え		
記録を停止	「記録を停止 (Stop	記録を開始したセッション。
	recording)]	

表 33. 開始、セッションの切り替え、および停止

現在選択されているセッション

現在選択されているセッションは、次のいずれかです。

- 記録が開始された最初のセッション (すべてのセッションで「記録を追加 (Append recording)」アイコンがクリックされていない場合)、または、
- 最後に「記録を追加 (Append recording)」アイコンがクリックされたセッション。

(2 番目の項目において、現在選択されているセッションが最初のセッションである 場合があります。別のセッションに切り替えた後に、最初のセッションに切り替え た場合です。) 現在選択されているセッションでは、「記録を追加 (Append recording)」アイコン は使用不可であり、そのセッションが、記録済みの素材の現在のソースであること を示しています。

現在選択されているセッションでは、「記録を休止 (Pause recording)」、「プロン プトを追加 (Add a Prompt)」、「スマート待機時間を追加 (Add a Smart Wait)」、および「抽出を追加 (Add an Extraction)」といったマクロ・マネージャ ー・アイコンがアクティブになっています。通常の記録を休止するときやエレメン トを追加する場合などに、これらのアイコンを使用します。

現在選択されているセッションが、最初に記録が開始されたセッションである場合 は、次のようになります。

- 「マクロを記録 (Record macro)」アイコンが使用不可になる。これは、そのセ ッションからマクロの記録が開始されたためです。
- 「記録を停止 (Stop recording)」アイコンが使用可能になる。これは、最初のセッションでは、記録を停止するためにこのアイコンがいつも使用可能になっているためです。

対照的に、現在選択されているセッションが記録を開始した最初のセッションでは ない場合は、次のようになります。

- 「マクロを記録 (Record macro)」ボタンが使用不可になる。これは、現在選択 されているセッションが他のマクロの記録済みの素材として使用されているとき に、そのセッションから新しいマクロの記録が開始されることを防止するためで す。
- 「記録を停止 (Stop recording)」ボタンが使用不可になる。これは、最初のセッションで開始された記録は、最初のセッションの「記録を停止 (Stop recording)」ボタン以外では停止することができないためです。

表 34 は、この情報をまとめたものです。

現住選択されているセッションの	状態:		
マクロ・マネージャー・アイコン:			
「記録を追加 (Append recording)」	使用不可		
• 休止	使用可能		
• 「プロンプトを追加 (Add a Prompt)」			
• 「スマート待機時間を追加 (Add a Smart			
Wait) J			
・ 「抽出を追加 (Add an Extraction)」			
記録	 現在選択されているセッションが最初のセッションの場合:使用不可 		
	 現在選択されているセッションが最初のセッションではない場合:使用不可 		
停止	 現在選択されているセッションが最初のセッションの場合:使用可能 		
	 現在選択されているセッションが最初のセッションではない場合:使用不可 		

表 34. 現在選択されているセッションのマクロ・マネージャー・アイコン

その他のアクティブ・セッション

セッションが現在選択されているセッションではなく、かつマクロの記録が開始さ れた最初のセッションでもない場合には、マクロ・マネージャー・ツールバーのす べてのアイコンは通常の状態にあり、正常に使用できます。

例えば、別のセッションで開始された別のマクロを記録している最中であったとしても、「マクロを記録 (Record macro)」アイコンをクリックすることにより、新しいマクロの記録を開始することができます。

複数のアクティブな記録

状況により、同時に複数のマクロを記録することが必要な場合があります。このような場合、独立したアクティブ・セッションでそれぞれのマクロを開始する必要があります。(マクロ・マネージャーは、「マクロを記録 (Record a macro)」アイコンをアクティブにし、アイコンがクリックされると記録を停止するまで選択できないようにすることによって、このルールを強制します。)それぞれのマクロは、切り離されて個別に記録されます。また、その記録が互いに干渉することはありません。

ただし、次の状態が発生することが想定されます。

- セッション A で開始したマクロを記録している。
- セッション B で開始した別のマクロを記録している。
- セッション C はアクティブであるが、記録は行っていない。
- セッション C からの素材を、セッション A で開始したマクロに記録する。

セッション C に記録済みの素材のソースを切り替えるには、セッション C の「記録を追加 (Append recording)」アイコンをクリックします。

この状態では、セッション C からの記録済みの素材を、セッション A で記録を開始したマクロに追加するのか、またはセッション B で記録を開始したマクロに追加 するのかが、マクロ・オブジェクトに伝わりません。したがって、セッション C の 「記録を追加 (Append recording)」アイコンをクリックすると、マクロ・オブジェ クトは、すぐに記録が開始されているセッション (セッション A およびセッション B) をリストしたウィンドウを表示し、セッション C からの記録済みの素材をどち らのセッションに追加するか、選択を求めます。

また、セッション C を使用して、セッション A に情報を記録している間は、セッション B に情報を記録する目的でセッション C を使用することはできません。

同一のマクロ画面での異なるセッションの参照

以前説明したマクロ画面の作成における第2の手法(193ページの『マクロ画面 作成の2つの手法』を参照)、つまり、同一のマクロ画面のディスクリプターやア クションで複数のセッションを参照する手法を使用する場合、マクロ・エディター またはコード・エディターを使用し、追加セッションへの参照を手動で追加する必 要があります。

例えば、3270 ディスプレイ・セッションと対話するマクロ画面を記録し、そのマク ロ画面に 5250 ディスプレイ・セッションへの参照を追加したい場合、マクロ・エ ディターまたはコード・エディターでそのマクロ・スクリプトを開き、 5250 ディ スプレイ・セッションに適切なディスクリプターやアクションを手動で追加する必 要があります。

次の 3 つのディスクリプターでは、マクロ・エディターの編集ウィンドウに表示で きるディスクリプターの数に制限があります。

- OIA が禁止解除になるのを待つ (Wait for OIA to Become Uninhibited)
- フィールド数 (Number of Fields)
- 入力フィールド数 (Number of Input Fields)

マクロ・エディターでは、これらの 3 つのディスクリプターは、単一のディスクリ プター、フィールド・カウント、および OIA ディスクリプターによって表されて います。さらに、通常のマクロでは、許されているフィールド・カウントおよび OIA ディスクリプターは、マクロ画面につき 1 つのみです (63 ページの 『「Field Counts and OIA」ディスクリプター』を参照)。通常のマクロにおける このような事情のため、それぞれのフィールド・カウントおよび OIA ディスクリ プターが異なるセッションを参照している場合でも、マクロ・エディターには複数 のフィールド・カウントおよび OIA ディスクリプターを表示する機能がありませ ん。したがって、マクロ画面に複数のフィールド・カウントおよび OIA ディスク リプターが含まれる場合、マクロ・エディターは最初の 1 つのみを表示します。た だし、コード・エディターでマクロ画面を表示する場合には、すべてのディスクリ プターを見ることができます。

マクロ・エディターを使用している場合、ディスクリプターで「現行 (Current)」ボ タンを使用し、マクロ・エディターが起動されたセッション以外のセッションから のデータを収集することができます (197 ページの『異なるセッションでの自動編 集機能の使用』を参照)。

マクロ記録のディセーブリング

ディセーブル機能のマクロ・ウィンドウには、複数のセッションと対話するマクロ の記録をイネーブルまたはディセーブルにする個別のオプションがありません。代 わりに、この機能はマクロの記録オプションの一部とみなされます。
第14章 グラフィカル・ユーザー・インターフェース

マクロ・エディター内のフィールドの更新

セッション・ウィンドウの使用

マクロ・エディター・ウィンドウはセッション・ウィンドウの前面に表示されます が、セッション・ウィンドウを使用することもできます。

操作するセッション・ウィンドウの領域が見えるように、マクロ・エディター・ウ ィンドウを画面の片側にドラッグします。その後、セッション・ウィンドウをクリ ックして現行ウィンドウにします (マクロ・エディターがセッション・ウィンドウ の一部にまだオーバーラップしていてもかまいません)。

マーキング長方形の使用

次のように、さまざまな状況でマーキング長方形を使用して、セッション・ウィン ドウの領域にマークを付けることができます。

- ストリング・ディスクリプター用のテキストの長方形ブロックにマークを付ける。
- 抽出アクションによってキャプチャーする領域にマークを付ける。
- ボックス選択アクションによってマーキング対象の領域にマークを付ける。
- 印刷抽出アクションによって印刷される領域にマークを付ける。

マーキング長方形を使用して領域にマークを付けるには、次の手順で行います。

- 操作するセッション・ウィンドウの領域が見えるように、マクロ・エディター・ ウィンドウを画面の片側にドラッグする。その後、セッション・ウィンドウをク リックします。
- マークを付けるセッション・ウィンドウの領域の片隅をマウスでクリックする。 テキスト・カーソルがその行と列の位置にジャンプすることを確認してください。
- 左マウス・ボタンを押したまま、マウスを移動する。マウスを動かすにつれて、 黄色いマーキング長方形の形状が変わります。
- 4. キャプチャーするテキストの領域を囲むようにマーキング長方形を調整してか ら、左マウス・ボタンを離す。
- 5. 黄色いマーキング長方形は、最も近い文字の行と列の境界に合わさります。
- 6. セッション・ウィンドウを再度クリックするまで、黄色いマーキング長方形はそ のまま表示されます。
- 7. 別の領域にマークを付ける場合は、前のステップ 2 からやり直してください。

セッション・ウィンドウのテキスト・カーソルの使用

アクションとディスクリプターによっては、行と列の座標の組を入力する必要が生 じることがあります。例を次に示します。 • 抽出アクションの「行 (Row)」と「列 (Column)」の入力フィールド。

テキスト・カーソルを使用してセッション・ウィンドウの行と列の位置を決定する には、次の手順で行います。

- 操作するセッション・ウィンドウの領域が見えるように、マクロ・エディター・ ウィンドウを画面の片側にドラッグする。その後、セッション・ウィンドウをク リックして現行ウィンドウにします (マクロ・エディターがセッション・ウィン ドウの一部にまだオーバーラップしていてもかまいません)。
- 2. 矢印キーを使用して、テキスト・カーソルを目的の行と列の位置に移動する。
- 3. 行/列の形式で (例: 04/17)、セッション・ウィンドウの右下隅に行と列の番号 が表示されます。
- 4. マクロ・エディター・ウィンドウをクリックして現行ウィンドウにする。
- 5. 行の値 (例: 4) を「行 (Row)」入力フィールドに入力し、列の値 (例: 17) を 「列 (Column)」入力フィールドに入力する。

ストリングを指定する際のエラー

ストリングを必要とする入力フィールドには、マクロに対して選択した形式 (基本 マクロ形式または拡張マクロ形式)の要求に沿った方法でストリングを指定する必 要があります (33 ページの『ストリングと特殊文字の表記、演算子文字の取り扱 い』を参照)。

例えば、拡張マクロ形式を選択した場合に、ストリングを単一引用符で囲まずに指 定すると (例: Terminal parameters)、マクロ・エディターは次のようなエラー・メ ッセージを表示します。

String -- Invalid expression -- Resetting to previous value.

このエラー・メッセージが表示されないようにするには、ストリングを単一引用符 で囲んで指定します (例: 'Terminal parameters')。

一方、基本マクロ形式を選択した場合に、ストリングを単一引用符で囲んで指定す ると、エラー・メッセージは出されませんが、マクロ・エディターは単一引用符を ストリングの一部として扱います。

コード・エディターの使用

本書からコード・エディターへのスクリプトのコピー・アンド・ペ ースト

ここでは、本書からコード・エディターにマクロ・スクリプトをコピーする方法を 説明します。この説明では、<HAScript> で始まり </HAScript> で終わるマク ロ・スクリプト全体をコピーすることを前提とします。次の手順を実行してくださ い。

- 1. 3270 ディスプレイ・セッションを開始し、接続する。
- 2. スクリプトのホルダーとして使用する単純なマクロを記録する。
 - a. 「マクロの記録 (Record Macro)」をクリックする。

- b. 「マクロの記録 (Record Macro)」ウィンドウが開いたら、次の操作を行う。
 - 本書からコード・エディターへスクリプトをコピー・アンド・ペースト する。
 - 2) 「新規 (New)」をクリックする。
 - 3) 「名前 (Name)」フィールドに sample1 などの名前を入力する。
 - 4) 「OK」をクリックする。
- c. 3270 ディスプレイ・セッションの状況表示行に、「マクロの記録中 (Recording macro)」と表示されます。
- d. 「マクロの再生または記録を停止 (Stop playing or recording macro)」を クリックする。
- 3. 記録したマクロ・スクリプトを編集する。
 - a. 記録したマクロのファイル名が、ウィンドウ内でマクロ・マネージャー・ツ ールバーの左側に sample1.mac のように表示されます。
 - b. 「現行マクロのプロパティーの編集 (Edit current macro properties)」をク リックして、マクロ・エディターを開始する。
 - c. マクロ・エディターが表示されたら、次の操作を行う。
 - 「コード・エディター」をクリックして、コード・エディターを開始する。
 - 2) マウスを使用して、削除するコードの行にマークを付ける。
 - a) 削除対象としてマークを付ける行は、コード・エディターに貼り付 けるテキストの内容によって異なります。
 - b) ただしこの例では、マクロ・スクリプト全体をコード・エディター に貼り付けることを前提とします。
 - c) したがって、この例では、マウスを使用してコード・エディター内 の行すべてに削除対象としてマークを付けます。
 - 3) Delete キーを押して、マークを付けた領域を削除する。
 - ユーザーが通常行っている方法を使用して、本書からシステム・クリッ プボードにマクロ・スクリプトのテキスト全体をコピーする。
 - 5) コード・エディターをアクティブ・ウィンドウにする。
 - 6) Ctrl-v を使用してマクロ・スクリプトをコード・エディターに貼り付ける。
 - 7) 「OK」をクリックしてコード・エディターを閉じる。
 - d. 「保管して終了」をクリックしてマクロ・スクリプトを保管し、マクロ・エ ディターを閉じる。
- 4. 編集したマクロのファイル名が、ウィンドウ内でマクロ・マネージャー・ツール バーの左側に sample1.mac のように表示されます。
- 5. 「マクロの再生 (Play Macro)」をクリックしてマクロを実行する。

このマクロを編集したい場合は、マクロ・エディターまたはコード・エディターの どちらを使用しても編集できます。

第3部マクロ言語

第15章 マクロ言語の機能

XML の使用

Host On-Demand マクロ言語の XML 構文

Host On-Demand マクロは、Host On-Demand マクロ言語の XML エレメントを 使用して XML スクリプトに保管されます。ここでは XML のいくつかの規則につ いて説明し、Host On-Demand マクロ言語の例を示します。

- XML コードはエレメントで構成されます。Host On-Demand マクロ言語には、 約 35 の XML エレメントがあります。
- マクロ言語のエレメント名には大/小文字の区別はありません。ただし、エレメントの開始タグと終了タグの両方に大文字と小文字の同じ組み合わせを使用する必要があるという意味では、大/小文字の区別があります。以下はすべて正しいエレメント名です(省略符号 "..." は XML テキストには含まれず、エレメントに他のエレメントが含まれていることを示すためのものです)。

<screen></screen>	•••	
<screen></screen>	•••	
<screen></screen>	•••	

ただし通例として、マスター・エレメントは HAScript と入力し、その他のエレ メントはすべて小文字で入力します。

 次に示す Host On-Demand マクロ言語の例のように、それぞれの XML エレメ ントには開始タグと終了タグがあります。

```
<HAScript> ... </HAScript>
<import> ... </import>
<vars> ... </vars>
<screen> ... </screen>
```

オプションで、XML エレメントの開始タグと終了タグを1 つのタグに結合できます。このオプションは、XML エレメントが属性を指定するだけで、他のエレメントを含まない場合に便利です。例えば、次のとおりです。

```
<oia ... />
<numfields ... />
```

エレメントには、attribute_name="attribute_value"の形式の属性を指定できます。例えば、次のとおりです。

<oia status="NOTINHIBITED" optional="false" invertmatch="false"/>
<numfields number="80" optional="false" invertmatch="false"/>

空の二重引用符の対 (つまり、間に何も入らない 2 つの二重引用符) を使用する と、属性に値を設定しないことを指定できます。

<HAScript name="ispf_ex1" description="" timeout="60000" ... author="" ...>

</HAScript>

HTML とほぼ同じように、エレメントの開始タグと終了タグの間に他のエレメント全体を含めることができます。次の例では、<description> エレメントに、
 <oia> エレメントと <numfields> エレメントの 2 つのエレメントが含まれています。

```
<description>
  <oia status="NOTINHIBITED" optional="false" invertmatch="false">
   <numfields number="80" optional="false" invertmatch="false"/>
  </description>
```

コード・エディター

コード・エディターを使用して、マクロ・スクリプトの XML テキストを直接編集 できます (10 ページの『コード・エディター』を参照)。

コード・エディターとシステム・クリップボードの間で、テキストのカット・アンド・ペーストを行うことができます。コード・エディターと、他の XML エディターやテキスト・エディターとの間でテキストを転送できるので、この機能は非常に重要です。

エレメントの階層

211 ページの図 80 は、Host On-Demand マクロ言語の XML エレメントすべて の開始タグのリストです。このリストは、XML 構文の観点から見て有効なものでは なく、同じタイプのエレメントを複数指定できる個所を示していません。ただしこ のリストでは、他の XML エレメント内で指定できる XML エレメントをインデン トによって示しています。例えば、リストの最初にあるエレメント(<HAScript> エ レメント)はまったくインデントされていません。これは、このエレメントがマス ター・エレメントであり、他のエレメントをすべて含むことを示しています。 2 番 目のエレメント(<import> エレメント)は、<HAScript> エレメントの内側にあ り、<type> エレメントを含みます。以降も同様です。

<hascript></hascript>	スクリプト内の他のエレメントをすべて囲みます。
<import></import>	<type> エレメントのコンテナー。</type>
<type></type>	インポートしたデータ型 (Java クラス) を宣言します。
<vars></vars>	<create> エレメントのコンテナー。</create>
<create></create>	変数を作成し、初期化します。
<screen></screen>	1 つのマクロ画面に関する情報を含む画面エレメント。
<description></description>	ディスクリプターのコンテナー。
<attrib></attrib>	特定のフィールド属性を記述します。
<cursor></cursor>	カーソルの位置を記述します。
<customreco></customreco>	カスタム認識エレメントを参照します。
<numfields></numfields>	画面内のフィールドの数を記述します。
<numinputfields></numinputfields>	画面内の入力フィールドの数を記述します。
<string></string>	画面の文字ストリングを記述します。
<varupdate></varupdate>	変数に値を割り当てます。
<actions></actions>	アクションのコンテナー。
<boxselection></boxselection>	ホスト・アプリケーション画面に選択ボックスを描画します。
<commwait></commwait>	指定の通信状況が発生するまで待ちます。
<custom></custom>	カスタム・アクションを呼び出します。
<extract></extract>	ホスト・アプリケーション画面からデータをコピーします。
<else></else>	else 条件を挿入できます。
<filexfer></filexfer>	ファイルをアップロードまたはダウンロードします。
<if></if>	if 条件を挿入できます。
<input/>	ホスト・アプリケーションにキー・ストロークを送ります。
<message></message>	ユーザーにメッセージを表示します。
<mouseclick></mouseclick>	マウス・クリックをシミュレートします。
<pause></pause>	指定の時間だけ待ちます。
<perform></perform>	ユーザー提供の Java メソッドを呼び出します。
<playmacro></playmacro>	別のマクロを呼び出します。
<prompt></prompt>	ユーザーにプロンプトを出して情報の入力を促します。
<trace></trace>	トレース・レコードを書き出します。
<varupdate></varupdate>	変数に値を割り当てます。
<nextscreens></nextscreens>	<nextscreen> エレメントのコンテナー。</nextscreen>
<nextscreen></nextscreen>	有効な次のマクロ画面の名前を指定します。
<recolimit></recolimit>	認識限度に達した場合にアクションを行います。

図 80. Host On-Demand マクロ言語のエレメントの階層

エレメントの階層と、対応するマクロ・スクリプトの構造については、本書のさま ざまな個所で説明しています。特に、次のセクションを参照してください。

- <HAScript> エレメントについては、 20 ページの『マクロ・スクリプトの概念 視点』を参照。
- <screen> エレメントについては、 26 ページの『マクロ画面の概念視点』を参照。

個々のエレメントの説明については、 219 ページの『第 16 章 マクロ言語エレメ ント』を参照してください。

マクロ・スクリプトへのコメントの挿入

XML スタイルのコメント・ブラケット <!-- --> を使用してコメントのテキストを 囲むことにより、<HAScript> エレメント内の任意の個所にコメントを挿入できま す。

コメントは、次の用途に便利です。

- 説明文を付けてマクロ・スクリプトを整理する。
- 複雑な事項の説明によりマクロ・スクリプトを文書化する。

 マクロ・スクリプトをデバッグする際に、エレメントのどの部分が問題の原因に なっているか判別するために、実行可能なエレメントをコメント化する。

コメントの形式

マクロ・スクリプトを保管する際に、コメントが次の形式に準拠するように、コード・エディターは必要に応じてコメントの形式を修正します。

- 各コメントは新しい行から始まる。
- 各コメントは後続のエレメントと同じスペースの数だけインデントされる。

コメントをどこに入れても、コード・エディターはこの方式に従ってコメントを整 列します (『コメントの例』を参照)。

コメント・エラー

コード・エディターは、次の状況が発生するとエラー・メッセージを表示します。

- コメントがネストしている
- コメントが実行可能エレメントの一部をコメント化している

また、<HAScript> エレメントの外側でブラケット <!-- --> を使用することはで きません。このようにした場合、コード・エディターはスクリプトを保管する際に これらのコメント・ブラケットと囲まれたテキストを破棄します。

コメントの例

次に、コメント・ブラケット <!-- --> を使用してコメントを挿入する例をいくつ か示します。

```
<!--
A multi-line comment that comments on
the following <screen> element
-->
<screen name="Screen1" entryscreen="true" exitscreen="false" transient="false">
<!-- A comment on the following <description> element -->
<description>
   <oia status="NOTINHIBITED" optional="false" invertmatch="false" />
</description>
<! A comment on the following <actions> element -->
<actions>
   <mouseclick row="4" col="16" />
  <input value="3[enter]" row="0" col="0" movecursor="true"
             xlatehostkeys="true" />
</actions>
<!--
BEGIN
An accidental comment that surrounds part of
a <nextscreens> element, thereby corrupting
the macro script.
You will get an error when you try to save
this macro script
<nextscreens timeout="0" >
  <nextscreen name="Screen2" />
END of accidental comment
-->
</nextscreens>
</screen>
```

<trace> エレメントを使用したマクロ・スクリプトのデバッグ

デバッグの際に <trace> エレメントを使用して、テキストと値をトレース出力先に 送ることができます。特に、出力に変数の名前を組み込むと、マクロ・ランタイム は変数の名前と値の両方を中括弧 {} で囲んで出力に表示します。次に例を示しま す。

```
<vars>
<create name="$var1$" type="string" value="'original'" />
</vars>
.
.
<actions>
<trace type="SYSOUT" value="'Before update: '+$var1$" />
<varupdate name="$var1$" value="'updated'" />
<trace type="SYSOUT" value="'After update: '+$var1$" />
</actions>
```

図 81. <trace> エレメントの使用例

前の図に示したコードは、次のテキストを Java コンソールに出力します。

```
Before update: +{$var1$ = original}
After update: +{$var1$ = updated}
```

図 82. <trace> エレメントの使用例の出力

<trace> アクションが、それぞれの変数の変数名と変数の内容を両方とも中括弧 {} で囲んで表示している点に注意してください。

マクロと組み合わせた Host Access Toolkit 製品の使用

別製品の Host Access Toolkit に組み込まれているクラスを使用して、マクロ変数 の動的作成、マクロ・アクションの実行、およびマクロの実行が可能です。ここで は、Host Access Toolkit 製品の使用例を示します。

214 ページの図 83 は、ユーザーの ID とパスワードのプロンプトを出し、ホスト にログオンして Welcome! と表示する、マクロの 1 つ目のバージョンを示してい ます。このバージョンのマクロは Host Access Toolkit を使用しません。

```
<HAScript name="Logon" description="" timeout="60000" pausetime="300"</pre>
          promptall="true" author="" creationdate="" supressclearevents="false"
          usevars="true" >
  <screen name="Screen1" entryscreen="true" exitscreen="false" transient="false">
    <description>
      <oia status="NOTINHIBITED" optional="false" invertmatch="false" />
    </description>
    <actions>
      <prompt name="'UserID:'" description="" row="20" col="16" len="8"</pre>
              default="" clearfield="false" encrypted="false" movecursor="true"
              xlatehostkeys="true" assigntovar="" varupdateonly="false" />
      <input value="'[tab]'" row="0" col="0" movecursor="true"
              xlatehostkeys="true" encrypted="false" />
      <prompt name="'Password:'" description="" row="21" col="16" len="8"</pre>
              default="" clearfield="false" encrypted="true" movecursor="true"
              xlatehostkeys="true" assigntovar="" varupdateonly="false" />
      <input value="'[enter]'" row="0" col="0" movecursor="true"
              xlatehostkeys="true" encrypted="false" />
    </actions>
    <nextscreens timeout="0" >
      <nextscreen name="Screen2" />
    </nextscreens>
  </screen>
  <screen name="Screen2" entryscreen="false" exitscreen="true" transient="false">
    <description>
      <oia status="NOTINHIBITED" optional="false" invertmatch="false" />
      <numfields number="7" optional="false" invertmatch="false" />
      <numinputfields number="1" optional="false" invertmatch="false" />
    </description>
    <actions>
      <message title="" value="'Welcome!'" />
    </actions>
    <nextscreens timeout="0" >
    </nextscreens>
  </screen>
```

```
</HAScript>
```

図 83. ユーザーの ID とパスワードのプロンプトを出すサンプル・マクロ

このマクロを Host Access Beans プログラムで使用し、ユーザー ID を変数に格 納して、後で使用する (例えば、Welcome メッセージの中で) ために保管したいと します。この用途のために直接マクロを変更することもできますが、このためのプ ログラムを作成する理由の 1 つは、さまざまな状況に応じて多数の異なるマクロを 保持しなくて済むようにすることです。代わりに、マクロの基本バージョンを作成 し、プログラムを使用してマクロを状況に応じて変更できます。次に、Java でこの 変更を行う方法の例を示します。

// Assume macro is an instantiated Macro with the appropriate listeners set up. // (See the Javadoc for the Macro bean and the Macro variables demo program, // MacroVariablesDemo.java, in the Host Access Toolkit samples directory // for details.) // Assume macroString is a String containing the previous macro script macro.setMacro(macroString); MacroScreens ms = macro.getParsedMacro(): ms.createVariableString("\$userid\$", null); //creates a variable \$userid\$ with //initial value of "" MacroScreen mscrn = ms.get(0); //get the first screen MacroActions mas = mscrn.getActions(); //get the actions from the first screen MacroActionPrompt map = (MacroActionPrompt)mas.get(0); //get the first prompt action map.setAssignToVar("\$userid\$"); //assign the prompt response to the variable \$userid\$ MacroScreen mscrn2 = ms.get(1); //get the second screen MacroActions mas2 = mscrn2.getActions(); //get the actions from the second screen MacroActionMessage mam = (MacroActionMessage)mas2.get(0); //get the message actionmam.setMessage("'Welcome ' + \$userid\$ + '!'"); //change the message to now be a //personalized message using \$userid\$ macro.setParsedMacro(ms); //reset the macro with the updated MacroScreens macro.play(); //play the macro with the changes for variables

図 84. 変数更新アクションとプロンプト・アクションを変更する Java コード

ここで、Screen2 のアクションに 2 つ目のメッセージを追加したいとします。この メッセージには、画面から抽出した時刻と日付を表示します。この場合、 macro.setParsedMacro(ms) の前に次の行を追加します。

//create a variable \$datetimestamp\$ with initial value ""
ms.createVariableString("\$datetimestamp\$", null);

//create new extract to get date and time from second row of screen
MacroActionExtract mae = new MacroActionExtract(2, 35, 2, 71, "'datetimeextract'");

//assign the date and time string to \$datetimestamp\$
mae.setAssignToVar("\$datetimestamp\$");

//add the extract after the first message
mas2.add(mae);

//create a new message to display the date and timestamp
MacroActionMessage mam2 = new MacroActionMessage(
 "'You have logged on at ' + \$datetimestamp\$", "'Date Time Stamp'");

//add the message after the extract
mas2.add(mam2);

図 85.2 つ目のメッセージの追加

変数を含む属性が MacroScreens に関連付けられる時点で、変数はすでに作成済み であることが必要なので注意してください (createVariable() メソッドの 1 つを使 用して)。例えば、次のコード・シーケンスも有効です。

```
MacroActionExtract mae = new MacroActionExtract(2, 35, 2, 71, "'datetimeextract'");
mae.setAssignToVar("$datetimestamp$");
ms.createVariableString("$datetimestamp$", null);
mas2.add(mae);
MacroActionMessage mam2 = new MacroActionMessage("'You have logged on at ' +
    $datetimestamp$", "'Date Time Stamp'");
mas2.add(mam2);
```

図 86. 代替コード・シーケンス

MacroActionExtract が MacroActions に追加される前に \$datetimestamp\$ が作成 されるので、前記のシーケンスは有効です (MacroActions は、元は MacroScreens から取り出されたものなので、すでに MacroScreens と関連付けられています)。前 記のシーケンスの最後に createVariable() メソッドが呼び出される場合、シーケン スは無効になります。これは、MacroActionExtract と MacroActionMessage が MacroActions に追加されて MacroScreens に関連付けられた時点では、変数 \$datetimestamp\$ は使用可能でないからです。

MacroScreens メソッド isUseVars() のデフォルト値は false です。ただし、 MacroScreens に対して createVariable() メソッドの 1 つを呼び出した場合、 isUseVars() は自動的に true を戻します。変数を作成せず、属性から変数と算術式 がスキャンされるようにしたい場合は (例えば、チェーニングした子マクロを作成 していて、子マクロが自身の変数を持たず、親からの変数を期待している場合)、 MacroScreens に対して setUseVars(true) を呼び出す必要があります。

変数または式を引数にすることができる属性には、 setAttribute(String) メソッド と、getAttributeRaw() または isAttributeRaw() のどちらかのメソッドを使用でき ます。式を使用して MacroActionInput の行属性を表したい場合は、 setRow("\$rowvar\$ + 1") を呼び出すことができます。その後、getRow() を呼び出 すとこの式の評価値 (整数) が戻されます。一方、getRowRaw() を呼び出すと "\$rowvar\$ + 1" が戻されます。次のようにすると、NumberFormatException が出 されることに注意してください。

```
MacroActionInput mai = new MacroActionInput();
mai.setRow("$rowvar$ + 1");
int row = mai.getRow();
```

図 87. NumberFormatException を引き起こすコード

これは、isUseVars() が true になる MacroScreens に mai がまだ関連付けられて いないからです。このため、"\$rowvar\$ + 1" は変数 + 1 ではなくストリングとし て扱われます。もう 1 つの注意点として、setAttribute() メソッドの呼び出しによ る変数と式のセットアップは、これらの属性を含むオブジェクトが MacroScreens に関連付けられた後で行えば、処理時間の節約が期待できます。このようにしない と、属性が MacroScreens に追加された時点で、属性の変数/式を再度構文解析する 必要が生じます。

VariableException クラスを使用して、無効な式 (例: "45 *") や正しくない演算オペ ランド (例: "'3a' * 2") などの例外を検出できます。 プログラム式マクロ MacroVariablesDemo.java を使用するサンプル・プログラム が、Host Access Toolkit の samples ディレクトリーにあります。

第16章 マクロ言語エレメント

属性の指定

XML 要件

マクロ言語では、どの属性の値も二重引用符で囲む必要があります。例えば、次の <mouseclick> エレメントでは、row と col 属性の値が二重引用符で囲まれます。 <mouseclick row="4" col="51" />

属性値の拡張形式

前述のように、マクロが拡張形式であっても、マクロ・エディター内のすべての入 カフィールドで、ストリングが単一引用符 ('') で囲まれるわけではありません (33 ページの『ストリングと特殊文字の表記、演算子文字の取り扱い』を参照)。具 体的には、拡張形式は、マクロ・エディターの次のタブ上の入力フィールドのみに 影響を与えます。

- 「画面 (Screens)」タブの「記述 (Description)」タブ
- 「画面 (Screens)」タブの「アクション (Actions)」タブ
- 「変数 (Variables)」タブ

同様に、マクロ言語では、拡張形式によって影響を受けるこれらの入力フィールド のいずれかに対応する属性に、ストリング値を指定する場合、そのストリングを拡 張形式で入力する必要があります。例えば、<message> エレメントでは、マクロが 拡張形式である場合、両方の属性のストリングを単一引用符で囲む必要がありま す。

<message title="'Instructions'" value="'Check the java console'" />

しかし、属性が、拡張形式によって影響を受ける入力フィールドのいずれかに対応 しない場合、マクロが拡張形式であっても、単一引用符で囲まれた値を入力する必 要はありません。例えば、<screen> エレメントの name 属性は単一引用符で囲み ません。

<screen name="Screen1" entryscreen="true" exitscreen="true" transient="false" >
...
</screen>

マクロ言語エレメントについてのこの章の説明では、データ型を指定しないことに よって、こうした属性 (拡張形式によって影響を受けない属性) を示します。例え ば、<screen> エレメントの name 属性の記述は、「必須のストリング」ではな く、「必須」です。

型付きデータ

大部分の属性には、特定のデータ型 (ブール、整数、ストリング、倍精度、または インポート)が必要です。これらの属性には、マクロ・エディターと同じ規則が適 用されます。

- 基本マクロ形式または拡張マクロ形式を選択した結果(33ページの『マクロ形式の選択』を参照)。
- ストリングと特殊文字を表す規則、および演算子文字を扱う規則(33ページの 『ストリングと特殊文字の表記、演算子文字の取り扱い』を参照)。
- 等価エンティティーの規則 (41 ページの『等価』を参照)。
- データ型変換の規則(40ページの『自動データ型変換』を参照)。
- 算術演算子と式の規則 (37 ページの『算術演算子および式』を参照)。
- ストリング連結演算子の規則(38ページの『ストリング連結演算子(+)』を参照)。
- 条件演算子と論理演算子および式の規則(39ページの『条件演算子と論理演算 子および式』を参照)。
- 変数を表す規則(154ページの『「変数(Variables)」タブの概要』を参照)。
- インポートした変数でメソッドを呼び出す規則(172 ページの『Java メソッドの呼び出し』を参照)。

<actions> エレメント

概要

<actions> エレメント、<description> エレメント、および <nextscreens> エレメ ントは、<screen> エレメント内に存在する 3 つの基本構造エレメントです (26 ページの『マクロ画面の概念視点』を参照)。

<actions> エレメントには、マクロの再生時にマクロ・ランタイムが実行する、ア クションと呼ばれるエレメント (例えば、キー・ストロークのシミュレート、デー タの取り込みなど) が含まれます (77 ページの『第 8 章 マクロ・アクション』を 参照)。

属性

promptall

オプションのブール (デフォルトは false)。この属性が true に設定される と、マクロ・ランタイムは、<actions> エレメント内のアクションを実行す る前に、そのエレメント内の <prompt> エレメントに対するユーザー入力 を収集します。具体的には次のとおりです。

- 1. マクロ・ランタイムは、<actions> エレメントを検索して、そのエレメ ント内にある <prompt> エレメントを見付ける。
- マクロ・ランタイムは、すべての <prompt> エレメントのプロンプト をただちに表示する (すべてのプロンプトは 1 つのポップアップに結合 されます)。
- 3. マクロ・ランタイムは、すべてのポップアップ・ウィンドウのユーザー 入力を収集する。
- 4. マクロ・ランタイムは、<actions> エレメント内のすべてのエレメント を、通常どおりに順に実行する。

 マクロ・ランタイムは、<prompt> アクションに達すると、ユーザー入 力用のポップアップ・ウィンドウを表示するのではなく、上記のステッ プ 3 からの入力を使用して <prompt> アクションを実行する。

<HAScript> エレメントの promptall 属性は、1 つのマクロ内のすべての <prompt> エレメントに対して同じ機能を実行します (234 ページの 『<HAScript> エレメント』を参照)。

XML サンプル

<actions promptall="true"> ... </actions>

図 88. <actions> エレメントの例

<attrib> エレメント

概要

<attrib> エレメントは、行と列の位置、および 3270 または 5250 属性の値を指定 するディスクリプターです (74 ページの『属性ディスクリプター (<attrib> エレメ ント)』を参照)。

属性

plane 必須。属性が置かれているデータ・プレーン。有効な値は次のとおりです。

- FIELD_PLANE
- COLOR_PLANE
- DBCS_PLANE
- GRID_PLANE
- EXFIELD PLANE
- 上記のいずれかに評価される任意の式

value 必須。形式 0x37 の 16 進値。属性の値。

row 必須の整数。データ・プレーン内の属性の行の位置。

col 必須の整数。データ・プレーン内の属性の列の位置。

optional

オプションのブール (デフォルトは false)。 60 ページの『オプショナル』 を参照してください。

invertmatch

オプションのブール。 60 ページの『逆ディスクリプター』を参照。

hostid

オプションのストリング。74ページの『セッションの指定』を参照。

XML サンプル

図 89. <attribute> エレメントの例

<boxselection> エレメント

概要

<boxselection> エレメントは、セッション・ウィンドウ上でマーキング長方形をドローして、ユーザーがセッション・ウィンドウをクリックし、左マウス・ボタンを押さえたままマウスをドラッグして、マーキング長方形を作成するアクションをシミュレートします (81 ページの『ボックス選択アクション (<boxselection> エレメント)』を参照)。

属性

srow	必須の整数。	マーキング長方形の起点コーナーの行座標。

- scol 必須の整数。マーキング長方形の起点コーナーの列座標。
- erow 必須の整数。マーキング長方形の終了コーナーの行座標。
- ecol 必須の整数。マーキング長方形の終了コーナーの列座標。
- type オプション (デフォルト SELECT)。マーキング長方形をドローするには SELECT を選択し、マーキング長方形を除去するには DESELECT を選択 します。

hostid

オプションのストリング。82ページの『セッションの指定』を参照。

XML サンプル

<boxselection srow="6" scol="16" erow="7" ecol="73" type="SELECT" />

図 90. <boxselection> エレメントの例

<comment> エレメント

概要

<comment> エレメントは、<screen> エレメント内のサブエレメントとしてテキス ト・コメントを挿入します。制限事項は次のとおりです。

• <screen> エレメントの外で <comment> エレメントを使用できない。

- 同じ <screen> エレメント内で複数の <comment> エレメントを使用できない。
 これを行うと、コード・エディターは、最後のエレメントを除いて、<screen>
 エレメント内のすべての <comment> エレメントを破棄します。
- <screen> エレメント内のどこに <comment> エレメントを置いても、コード・ エディターは、コメントを上に移動して、<screen> エレメント内の最初のエレ メントにする。

コメントの挿入方法

コメントを挿入するもっと柔軟な方法は、XML スタイルのコメント大括弧 (<!---->) を使用することです。 211 ページの『マクロ・スクリプトへのコメントの挿 入』を参照してください。

属性

なし。

XML サンプル

図 91. <comment> エレメントの例

<commwait> エレメント

概要

<commwait> アクションは、セッションの通信状況が、指定されたなんらかの値に 変わるのを待機します (82 ページの『通信待機アクション (<commwait> エレメ ント)』を参照)。タイムアウト値を指定する必要があります。

属性

value 必須。待機する通信状況。この値は、次のいずれかでなければなりません (83 ページの『通信状態』を参照)。

- CONNECTION_INIT
- CONNECTION_PND_ACTIVE
- CONNECTION_ACTIVE
- CONNECTION_READY
- CONNECTION_DEVICE_NAME_READY
- CONNECTION_WORKSTATION_ID_READY
- CONNECTION_PND_INACTIVE

CONNECTION_INACTIVE

timeout

必須の整数。ミリ秒単位のタイムアウト値。指定された通信状況が発生しな いうちにタイムアウトになる場合、マクロ・ランタイムはアクションを終了 します。

hostid

オプションのストリング。84 ページの『セッションの指定』を参照。

XML サンプル

<commwait value="CONNECTION_READY" timeout="10000" />

図 92. <commwait> エレメントの例

<condition> エレメント

概要

<condition> エレメントは、マクロ・ランタイムが画面認識時に評価する条件式を 指定します。式が true に評価される場合、マクロ・ランタイムはこのディスクリ プターを true と評価します。式が false に評価される場合、マクロ・ランタイム はこのディスクリプターを false と評価します (75 ページの『条件ディスクリプ ター (<condition> エレメント)』を参照)。

条件式の詳細については、 39 ページの『条件演算子と論理演算子および式』を参照してください。

属性

value 必須式。マクロ・ランタイムが評価する条件式。この条件式には、演算式、 変数、Java メソッド呼び出しからの戻り値、およびその他の条件式を含む ことができます。

optional

オプションのブール (デフォルトは false)。 60 ページの『オプショナル』 を参照してください。

invertmatch

オプションのブール。 60 ページの『逆ディスクリプター』を参照。

図 93. <condition> エレメントの例

<create> エレメント

概要

<create> エレメントは変数を作成し、初期化します (157 ページの『変数の新規作 成』を参照)。

<create> エレメントは、<vars> エレメント内に存在する必要があります。

属性

- name 必須。変数に割り当てる名前。変数名のスペルには、いくつかの制限があり ます (160 ページの『変数名と型名』を参照)。
- type 必須。変数の型。標準の型は、ストリング、整数、倍精度、ブール、フィー ルドです。また、Java クラスを表すインポート型も定義できます (157 ペ ージの『変数の新規作成』を参照)。
- value オプション。変数の初期値。初期値を指定しない場合、デフォルトの初期値 は変数の型によって異なります (156 ページの表 22 を参照)。

```
<HAScript ... usevars="true" ... >
    <import>
        <type class="java.util.Properties" name="Properties" />
        </import>
        <vars>
            <create name="$prp$" type="Properties" value="$new Properties()$" />
            <create name="$strAccountName$" type="string" value="" />
            <create name="$intAmount$" type="integer" value="0" />
            <create name="$dblDistance$" type="double" value="0.0" />
            <create name="$boolSignedUp$" type="boolean" value="false" />
            <create name="$fldFunction$" type="field" />
            </vars>
            ...
</HAScript>
```

図 94. <create> エレメントの例

<cursor> エレメント

概要

<cursor> エレメントは、セッション・ウィンドウ上のテキスト・カーソルの行と列 の位置を指定するディスクリプターです (73 ページの『カーソル・ディスクリプ ター (<cursor> エレメント)』を参照)。

属性

row 必須の整数。テキスト・カーソルの行の位置。

col 必須の整数。テキスト・カーソルの列の位置。

optional

オプションのブール (デフォルトは false)。 60 ページの『オプショナル』 を参照してください。

invertmatch

オプションのブール。 60 ページの『逆ディスクリプター』を参照。

hostid

オプションのストリング。74ページの『セッションの指定』を参照。

XML サンプル

<cursor row="4" col="14" optional="false" invertmatch="false" />

図 95. <cursor> エレメントの例

<custom> エレメント

概要

<custom> エレメントを使用すると、マクロ画面の <actions> エレメント内から、 カスタム Java プログラムを起動することができます。ただし、別個の Host Access Toolkit 製品を使用する必要があります。

プロセスの概要は次のとおりです。

- 1. マクロ画面の <actions> エレメントの処理中にアクションとして起動したい Java プログラムがあるものとします。
- コード・エディターで、カスタム Java プログラムを起動するロケーションで、
 <actions> エレメントに次の行を追加します。
 <custom id="'MyProgram1'" args="'arg1 arg2 arg3'" />
- Host Access Toolkit 製品の MacroActionCustom クラスの指示に従います。 MacroCustomActionListener をインプリメントするクラスを作成します。マク ロ・ランタイムがステップ 2 の <custom> アクションを実行すると、execute() メソッドがイベントで呼び出されます。

属性

- id 必須。実行したいカスタム Java プログラムを識別する任意のストリング。
- args オプション。カスタム Java プログラムに渡したい引数。

XML サンプル

<custom id="'MyProgram1'" args="'arg1 arg2 arg3'" />
<custom id="'MyProgram2'" args="'arg1 arg2'" />

図 96. <custom> エレメントの例

<customreco> エレメント

概要

この <customreco> エレメントを使用すると、カスタム記述コードにコールアウト することができます。<customreco> エレメントを使用するには、別個の Host Access Toolkit 製品が必要です。

カスタム・ディスクリプターを作成する手順は、次のとおりです。

- カスタム記述を識別するストリング (例えば、MyCustomDescriptor01) を選択する。複数のタイプのカスタム記述があるので、ID は必須です。
- ECLCustomRecoListener インターフェースをインプリメントする。doReco() メソッドで、次のことを行います。

- a. 識別ストリングを調べるためのコードを追加して、そのストリングが自分の ものであることを確認する。
- b. カスタム記述コードを追加する。
- c. カスタム記述に適合する場合は true を戻し、適合しない場合は false を戻 す。
- コード・エディターを使用して、<customreco> エレメントをマクロ画面の
 <description> エレメントに追加する。<customreco> エレメントは、ステップ
 2 で選択した ID を指定する必要があります。

マクロ・ランタイムは、他のすべてのディスクリプターを実行した後、
<customreco> エレメントを実行します。

属性

id 必須のストリング。このカスタム記述に割り当てた ID。

optional

オプションのブール (デフォルトは false)。 60 ページの『オプショナル』 を参照してください。

- invertmatch
 - オプションのブール。 60 ページの『逆ディスクリプター』を参照。

XML サンプル

<customreco id="'MyCustomDescriptor01'" optional="false" invertmatch="false" />

図 97. <customreco> エレメントの例

<description> エレメント

概要

<actions> エレメント、<description> エレメント、および <nextscreens> エレメ ントは、<screen> エレメント内に存在する 3 つの基本構造エレメントです (26 ページの『マクロ画面の概念視点』を参照)。

<description> エレメントには、ディスクリプターと呼ばれるエレメントが含まれま す。各ディスクリプターは、アプリケーション画面の識別特性を指定します (53 ページの『第7章 画面記述と画面認識』を参照)。マクロ・ランタイムは、ディス クリプターを使用して、マクロ画面をアプリケーション画面と一致させます。

属性

uselogic

オプションのブール。デフォルト結合メソッドで使用可能な複数のディスク リプター間で、もっと複雑な論理関係を定義できます (61 ページの 『uselogic 属性』を参照)。

XML サンプル

<description uselogic="true">

</actions>

図 98. <description> エレメントの例

<else> エレメント

概要

<else> エレメントは、一連のマクロ・アクションを含み、<if> エレメントの直後に 存在しなければなりません。マクロ・ランタイムは、<if> エレメント内の条件式を 評価します。次に、

- 条件式が true である場合
 - マクロ・ランタイムは、<if>エレメント内のその一連のマクロ・アクション を実行する。
 - マクロ・ランタイムは、後続の <else> エレメントがあれば、これをスキップ する。
- 条件式が false である場合
 - マクロ・ランタイムは、<if>エレメント内のその一連のマクロ・アクション をスキップする。
 - マクロ・ランタイムは、後続の <else> エレメント内のマクロ・アクションが あれば、これを実行する。

マクロ・オブジェクトは、<if> エレメント、および必要に応じて <else> エレメン トを使用して、条件アクションを保管します (84 ページの『条件アクション (<if> エレメントおよび <else> エレメント)』を参照)。

属性

なし。

<if condition="(\$var_int\$ > 10)"> ... </if> <else> ... </else>

図 99. <else> エレメントの例

<extract> エレメント

概要

この <extract> アクションは、セッション・ウィンドウからデータを取り込みます (87 ページの『抽出アクション (<extract> エレメント)』を参照)。

属性

下記のすべての属性の使用については、 87 ページの『抽出アクション (<extract> エレメント)』を参照してください。

name 必須のストリング。抽出されたデータに割り当てられる名前。この名前は、 Host Access Toolkit 製品を使用する場合だけ有効です。

planetype

必須。データが取り出される元のプレーン。 TEXT_PLANE 以外のデー タ・プレーンにアクセスするには、Host Access Toolkit 製品が必要です (92 ページの『Toolkit を使用したデータ・プレーンからのデータのキャプ チャー』を参照)。有効な値は次のとおりです。

- TEXT_PLANE
- FIELD_PLANE
- COLOR_PLANE
- EXFIELD_PLANE
- DBCS_PLANE
- GRID_PLANE
- srow 必須の整数。行と列の座標の最初のペアの行。
- scol 必須の整数。行と列の座標の最初のペアの列。
- erow 必須の整数。行と列の座標の2番目のペアの行。
- scol 必須の整数。行と列の座標の2番目のペアの列。

unwrap

オプションのブール。この属性を true に設定すると、マクロ・ランタイム は、指定された長方形内で始まる任意のフィールドの内容全体を取り込みま す。 90 ページの『「テキストのアンラップ (Unwrap Text)」オプショ ン』を参照。 continuous

オプションのブール。この属性を true に設定すると、マクロ・ランタイム は、必要に応じて次の行に折り返す連続した一連のデータの開始と終了の位 置として、行と列の座標を解釈します。この属性を false に設定すると、マ クロ・ランタイムは、長方形のテキスト域の左上隅と右下隅として、行と列 の座標を解釈します。 90 ページの『セッション・ウィンドウからのテキス ト・シーケンスのキャプチャー』を参照。

assigntovar

オプションの変数名。この属性を変数名に設定すると、マクロ・ランタイム は、テキスト・プレーン・データをストリング値として変数に保管します。 変数が、ストリング以外の標準型(つまり、ブール、整数、または倍精度) である場合、可能であれば、データはその標準型に変換されます。データが 変換できない場合、マクロは終了し、ランタイム・エラーを出します(88 ページの『テキストを格納する変数を指定する』を参照)。

hostid

オプションのストリング。 93 ページの『セッションの指定』を参照。

XML サンプル

図 100. <extract> エレメントの例

<fileupload> エレメント

概要

<fileupload> エレメントはホスト・データベース内のテーブルの作成、置換、デー タの追加、または更新を行います (93 ページの『FileUpload アクション (<fileupload> エレメント)』を参照)。

属性

- url 必須のストリング。jdbc:as400://myISeries など、ファイル・アップロード・コマンドの送信先となるデータベース・サーバーのデータベース URL です (93 ページの『データベースの URL』を参照)。
- driver 必須のストリング。データベース・サーバーとの接続に使用されるドライバ ーの完全修飾パッケージ名です (COM.ibm.db2.jdbc.app.DB2DRIVER など)。 このパッケージは、クライアント・ワークステーション上に存在している必 要があります (94 ページの『ドライバー ID とドライバー・クラス』を参 照)。

userid

オプションのストリング。データベースにアクセスするためのユーザー ID (必要な場合)(95 ページの『ユーザー ID とパスワード』を参照)。

password

オプションのストリング。データベースにアクセスするためのパスワード (必要な場合)(95ページの『ユーザー ID とパスワード』を参照)。

filename

必須のストリング。ホスト・データベース内のテーブルに追加されるデータ を含むローカル・ファイルの完全パスおよび名前です(96ページの『ファ イル名およびファイル・タイプ』を参照)。

filetype

必須の整数。ホスト・データベース内のテーブルに追加されるデータを含む ローカル・ファイルのタイプです (96 ページの『ファイル名およびファイ ル・タイプ』を参照)。有効な値は次のとおりです。

- 0 ASCII テキスト (*.txt)
- 1 コンマ区切り値 (*.csv)
- 2 Lotus 1-2-3 (*.wk1)
- 3 Microsoft Excel BIFF3 (*.xls)
- 4 Microsoft Excel BIFF4 (*.xls)
- 5 XML (*.xml)

uploadtype

必須のストリング。実行するファイル・アップロード処理のタイプです。有 効な値は次のとおりです。

- append ホスト・データベース内の既存のテーブルに行を追加します (97 ページの『付加』を参照)。
- create ホスト・データベース内に新規テーブルを作成します(96 ページの『作成』を参照)。このアップロード・タイプを使用する場合は、以下の属性も指定する必要があります。
 - fielddesctable
- replace ホスト・データベース内の既存のテーブルの内容を置き換えます (96 ページの『置換』を参照)。
- update 既存のテーブルの一部を選択的に更新します(97 ページの 『更新』を参照)。このアップロード・タイプを使用する場合は、以下の 属性も指定する必要があります。
 - keycolumns

fielddesctable

ストリング (uploadtype が create の場合に必須)。データベース・サーバ ーが新規テーブルの列名および列幅を読み取るホスト・データベース内のテ ーブル名です (96 ページの『作成』を参照)。

keycolumns

ストリング (**uploadtype** が update の場合に必須)。更新する列の名前 (複 数可) です (97 ページの『更新』を参照)。

```
<fileupload url="'jdbc:as400://elcrtp06'"
driver="'com.ibm.as400.access.AS400JDBCDriver'"
userid="'myuser'"
password="Ex0bRtrf73mPrwGrWMT+/g=="
filename="e:¥¥tm¥¥db02.txt"
filetype="4"
table="'hod.hodtest01'"
uploadtype="append" />
```

図 101. <fileupload> エレメントの例

<filexfer> エレメント

概要

<filexfer> アクションは、ワークステーションからホストへ、またはホストからワ ークステーションへ、ファイルを転送します (87 ページの『抽出アクション (<extract> エレメント)』を参照)。

属性

direction

必須。ワークステーションからホストへファイルを転送するには、send を 使用し、ホストからワークステーションへファイルを転送するには receive を使用してください。

pcfile 必須のストリング。ワークステーション上のファイルの名前(134 ページの『基本パラメーター』を参照)。

hostfile

必須のストリング。ホスト上のファイルの名前 (134 ページの『基本パラ メーター』を参照)。

clear 必須のブール。3270 ディスプレイ・セッションの場合は true に設定し、 5250 ディスプレイ・セッションの場合は false に設定してください (134 ページの『高度なパラメーター』を参照)。

timeout

必須の整数。ミリ秒単位のタイムアウト値 (デフォルト値は 10000 ミリ 秒)。ファイルが転送されないうちにこのタイムアウトになると、マクロ・ ランタイムは転送を終了します。

options

オプションのストリング。ホスト・システムが必要とする任意の追加パラメ ーター。

pccodepage

オプションの整数 (例えば、437)。ワークステーションの文字セットから、 ホストの文字セットへの文字のマッピング、およびその逆の文字のマッピン グで使用するローカル・コード・ページです。デフォルト値は、セッション 構成で指定されるコード・ページです。

hostorientation

オプション。BIDI セッション専用 (アラビア語とヘブライ語)。ホスト・フ ァイルのテキスト方向が、右から左であるか、左から右であるかを指定しま す。

pcorientation

オプション。BIDI セッション専用 (アラビア語とヘブライ語)。ローカル・ ファイルのテキスト方向が、右から左であるか、左から右であるかを指定し ます。

pcfiletype

オプション。BIDI セッション専用 (アラビア語とヘブライ語)。ローカル・ ファイル・タイプが可視であるか、暗黙であるかを指定します。

lamalefexpansion

オプションのブール。BIDI セッション専用 (アラビア語のみ)。ラームとア リフ拡張がオンであるかどうかを指定します。

lamalefcompression

オプションのブール。BIDI セッション専用 (アラビア語のみ)。ラームとア リフ圧縮がオンであるかどうかを指定します。

hostid

オプションのストリング。 135 ページの『セッションの指定』を参照。

XML サンプル

図 102. <filexfer> エレメントの例

<HAScript> エレメント

概要

<HAScript> エレメントは、マクロ・スクリプトのマスター・エレメントです。そ の他のエレメントを含み、マクロについてのグローバル情報を指定します (20 ペ ージの『マクロ・スクリプトの概念視点』を参照)。

属性

name 必須。マクロの名前。マクロ名は大/小文字が区別されます。

description

オプション。このマクロについての記述テキスト。このマクロについて覚え ておきたい情報をここに組み込みます。

timeout

オプションの整数。画面認識に使用できるミリ秒数。このタイムアウト値が 指定されているときに、これを超えると、マクロ・ランタイムは、マクロを 終了し、メッセージを表示します (142 ページの『画面間のタイムアウト (「マクロ (Macro)」タブ)』を参照)。デフォルトでは、マクロ・エディター はこの値を 60000 ミリ秒 (60 秒) に設定します。

pausetime

オプションの整数。"アクション間の休止"の遅延です(145 ページの『ア クション間の休止(「マクロ (Macro)」タブ)』を参照)。デフォルトでは、 マクロ・エディターはこの値を 300 ミリ秒に設定します。

promptall

必須のブール。この属性が true に設定されると、マクロ・ランタイムは、 最初のマクロ画面でアクションを実行する前に、マクロ全体内のすべての <prompt> エレメントに対するユーザー入力を収集し、個々のプロンプトを 結合して 1 つの大きなプロンプトにします。 <actions> エレメントの promptall 属性は、1 つの <actions> エレメント内のすべての <prompt> エレメントに対して類似した機能を実行します (220 ページの『<actions> エレメント』を参照)。

author

オプション。このマクロの作成者 (単数または複数)。

creationdate

オプション。このマクロの日付とバージョンについての情報。

suppressclearevents

オプションのブール (デフォルトは false)。ホスト・アプリケーションが、 clear screen コマンドの直後に、データ・ストリーム内のレコード終わり標 識を続けて送信するときに、システムが画面イベントを無視するかどうかを 判別する拡張機能。アプリケーション・フロー内の画面に、すべてブランク のものがある場合、この値を true に設定できます。マクロ内に有効なブラ ンク画面があるときに、clear コマンドが無視される場合、すべてのブラン クの画面イベントが、動作不良のホスト・アプリケーションからの clear コ マンドによって生成される可能性があります。これにより、画面認識イベン トが処理され、有効なブランク画面が、一致してはならないときに一致しま す。

usevars

必須のブール (デフォルトは false)。この属性が true に設定される場合、 マクロは拡張マクロ形式を使用します (33 ページの『マクロ形式の選択』 を参照)。

ignorepauseforenhancedtn

オプション。3270 ディスプレイ・セッションのみ。この属性が true に設 定される場合、セッションが、コンテンション解消モードで実行される TN3270E セッションであれば、マクロ・ランタイムは、すべての <pause> エレメントをスキップします (150 ページの『画面の完了に関係する属 性』を参照)。特定の <pause> エレメントを再度使用可能にするには、<pause> エレメントの ignorepauseoverride 属性を参照してください。

delayifnotenhancedtn

オプション。3270 ディスプレイ・セッションのみ。この属性は、ミリ秒単 位で値を指定し、セッションがコンテンション解消モードで実行される TN3270E セッションでない 場合だけ効果があります。その状態では、この 属性により、マクロ・ランタイムは、OIA 標識が変更されたという通知を 受け取るたびに、指定された期間の休止を追加します (150 ページの『画 面の完了に関係する属性』を参照)。

blockTerminalInput

オプション。この属性の値が true のときは、マクロの再生中、キーボード 入力およびマウス・クリックは無視されます。イベントは廃棄されます。入 力がブロックされたことを表示するメッセージ・アクションを作成しないか ぎり、ユーザーに対する表示はありません。

ignorepausetimeforenhancedtn

コンテンション解消環境での実行時にマクロの pausetime 属性を無視でき るようになりました。新規属性の「ignorepausetimeforenhancedtn」が、 <HAScript> マクロ・エレメントに追加されました。この属性では、 <HAScript> の pausetime 属性、および <screen> の pause 属性の両方を 無視できます。この属性に指定できる値は、次のとおりです。

True: <HAScript> の pausetime 属性と <screen> の pause 属性の両方が 無視されます。入力/プロンプト・アクションの後、または画面間で、一時 停止は発生しません。これはデフォルト値です。

False <HAScript> の pausetime 属性、および <screen> の pause 属性は 無視されません。

ignorepausetimeforenhancedtn は、非コンテンション解消環境で実行され ている場合は、影響がないことに注意してください。つまり、

ignorepausetimeforenhancedtn が true でも、非コンテンション解消環境で 実行されている場合は、<HAScript> の pausetime 属性、および <screen> の pause 属性は無視されません。

コンテンション解消環境でのマクロの実行方法に影響を与える、

ignorepauseforenhancedtn、delayifnotenhancedtn、およびその他 2 つの属 性について詳しくは、「マクロ・プログラミング・ガイド」を参照してくだ さい。 ignorepausetimeforenhancedtn の値を表示または変更するには、マ クロ・エディターのコード・エディターを使用してください。

</HAScript>

図 103. <HAScript> エレメントの例

<if> エレメント

概要

<if> エレメントには、条件式と一連のマクロ・アクションが含まれています。マク ロ・ランタイムは、<if> エレメント内の条件式を評価します。次に、

- 条件式が true である場合
 - マクロ・ランタイムは、<if>エレメント内のその一連のマクロ・アクション を実行する。
 - マクロ・ランタイムは、後続の <else> エレメントがあれば、これをスキップ する。
- 条件式が false である場合
 - マクロ・ランタイムは、<if>エレメント内のその一連のマクロ・アクション をスキップする。
 - マクロ・ランタイムは、後続の <else> エレメント内のマクロ・アクションが あれば、これを実行する。

マクロ・オブジェクトは、<if> エレメント、および必要に応じて <else> エレメン トを使用して、条件アクションを保管します (84 ページの『条件アクション (<if> エレメントおよび <else> エレメント)』を参照)。

属性

condition

必須。条件式。条件式には論理演算子と条件演算子を指定でき、また演算 式、即時値、変数、および Java メソッドの呼び出しを含む項を指定できま す (39 ページの『条件演算子と論理演算子および式』を参照)。

```
<vars>
   <create name="$condition1$" type="string"/>
   <create name="$condition2$" type="boolean" value="false"/>
   <create name="$condition3$" type="integer"/>
</vars>
<screen>
   <description>
         . . .
   </description>
   <actions promptall="true">
      <extract name="Get condition 1" srow="2" scol="1" erow="2"</pre>
                ecol="80" assigntovar="$condition1$"/>
      <extract name="Get condition 2" srow="3" scol="1" erow="3"
        ecol="80" assigntovar="$condition2$"/>
      <extract name="Get condition 3" srow="4" scol="1" erow="4"</pre>
                ecol="80" assigntovar="$condition3$"/>
      <if condition=
                "(($condition1$ !='')&&
                ($condition2$) || ($condition3$ < 100)) ">
      </if>
      <else>
      </else>
   </actions>
</screen>
```

図 104. <if> エレメントの例

<import> エレメント

概要

<import> エレメント、<vars> エレメント、および <screen> エレメントは、<HAScript> エレメント内に存在する 3 つの基本構造エレメントです (20 ページの『マクロ・スクリプトの概念視点』を参照)。

<import> エレメントはオプションです。これには <type> エレメントが含まれ、 それぞれが、Java クラスに基づいてインポート型を宣言します (158 ページの 『Java クラスのインポート型の作成』を参照)。

<import> エレメントは、<HAScript> 開始タグの後、かつ <vars> エレメントの前 に存在しなければなりません。

属性

なし。
```
<HAScript .... >
    <import>
        <type class="java.util.Properties" name="Properties" />
        </import>
        <vars>
            <create name="$prp$" type="Properties" value="$new Properties()$" />
            </vars>
        ...
        </HAScript>
```

図 105. <import> エレメントの例

<input> エレメント

概要

<input> エレメントは、キー・ストロークのシーケンスをセッション・ウィンドウ に送信します。このシーケンスには、文字 (例えば、a、b、c、#、& など) を表示 するキー、およびアクション・キー (例えば、[enterreset]、[copy]、[paste] など) を含むことができます (97 ページの『入力アクション (<input> エレメント)』を 参照)。

属性

- **value** 必須のストリング。セッション・ウィンドウに送信される一連のキー (98 ページの『入力ストリング』を参照)。
- row オプションの整数 (デフォルトは、テキスト・カーソルの現在位置)。入力が 開始する行 (97 ページの『タイプ入力の開始位置』を参照)。
- col オプションの整数 (デフォルトは、テキスト・カーソルの現在位置)。入力が 開始する桁 (97 ページの『タイプ入力の開始位置』を参照)。

movecursor

オプションのブール (デフォルトは true)。この属性を true に設定する と、マクロ・ランタイムは、テキスト・カーソルを入力の末尾に移動します (99 ページの『カーソルを入力の最後に移動』を参照)。

xlatehostkeys

オプションのブール (デフォルトは true)。この属性を true に設定する と、マクロ・ランタイムは、アクション・キー (例えば、[enter])の名前 を、文字シーケンスとしてではなく、アクション・キーとして解釈します (99 ページの『ホスト・アクション・キーの変換』を参照)。

encrypted

オプションのブール (デフォルトは false)。この属性を true に設定する と、コード・エディターは、終了時に value 属性に格納されている一連の キーを暗号化します (99 ページの『パスワード』を参照)。

hostid

オプションのストリング。 104 ページの『セッションの指定』を参照。

図 106. <input> エレメントの例

<message> エレメント

概要

<message> エレメントは、表題、メッセージ、および「OK」ボタンを含む、ポッ プアップ・ウィンドウを表示します。マクロ・ランタイムは、ユーザーが OK をク リックするまで待ってから、次のアクションに進みます (104 ページの『メッセー ジ・アクション (<message> エレメント)』を参照)。

属性

title オプションのストリング (デフォルトはマクロ名)。ポップアップ・ウィンド ウの表題バーに表示されるストリング。

value 必須のストリング。ポップアップ・ウィンドウに表示されるメッセージ。

XML サンプル

<message title="'Ready'" value="'Ready to process. Click OK to proceed.'" />

図 107. <message> エレメントの例

<mouseclick> エレメント

概要

<mouseclick> エレメントは、ユーザーによるセッション・ウィンドウ上のマウス・ クリックをシミュレートします。実際のマウス・クリックと同じように、テキス ト・カーソルは、クリックが行われたときにマウス・アイコンがポイントしていた 行と列の位置にジャンプします (105 ページの『マウス・クリック・アクション (<mouseclick> エレメント)』を参照)。

属性

- row 必須の整数。マウス・クリックが行われるセッション・ウィンドウ上の行と 列の位置の行。
- col 必須の整数。マウス・クリックが行われるセッション・ウィンドウ上の行と 列の位置の列。

hostid

オプションのストリング。105 ページの『セッションの指定』を参照。

XML サンプル

<mouseclick row="20" col="16" />

図 108. <mouseclick> エレメントの例

<nextscreen> エレメント

概要

<nextscreen> エレメントは、マクロ・ランタイムが、特に次に処理するマクロ画面 の候補と見なす <screen> エレメント (マクロ画面) の名前を指定します (137 ペ ージの『有効な次画面』を参照)。

<nextscreen> エレメントは、<nextscreens> エレメント内に存在する必要があります。

属性

name 必須。次に処理するマクロ画面の候補である <screen> エレメントの名前。

XML サンプル

```
<!--
The effect of the following <nextscreens> element and its contents
is that when the macro runtime finishes performing the actions in
the current screen, it adds ScreenS and ScreenG to the runtime list of
valid next screens.
    -->
    <nextscreens>
        <nextscreen name="ScreenS">
            <nextscreen name="ScreenS">
            </nextscreens>
        </nextscreens>
```

<nextscreens> エレメント

概要

<actions> エレメント、<description> エレメント、および <nextscreens> エレメ ントは、<screen> エレメント内に存在する 3 つの基本構造エレメントです (26 ページの『マクロ画面の概念視点』を参照)。

<nextscreens> エレメントには、複数の <nextscreen> エレメントが含まれ、各エレ メントは、現行のマクロ画面の後に表示されるマクロ画面の名前を指定します (137 ページの『第 9 章 画面認識、パート 2』を参照)。

属性

timeout

オプションの整数。画面認識タイムアウトの値 (ミリ秒)。マクロ・ランタイ ムは、このタイムアウト値が満了する前に、名前が有効な次画面のランタイ ム・リスト上にあるマクロ画面を、アプリケーション画面と一致させること ができない場合、マクロを終了します (142 ページの『画面認識のタイム アウト設定』を参照)。

XML サンプル

```
<!--

The effect of the following <nextscreens> element and its contents

is that when the macro runtime finishes performing the actions in

the current screen, it will attempt to recognize ScreenS and ScreenG.

-->

<nextscreens>

<nextscreen name="ScreenS">

<nextscreen name="ScreenS">

</nextscreens>
```

<numfields> エレメント

概要

<numfields> エレメントは、セッション・ウィンドウ内に存在する、すべての型の 3270 または 5250 フィールドの数を指定するディスクリプターです (67 ページの 『「フィールド数 (Number of Fields)」ディスクリプター (<numfields> エレメン ト)』を参照)。

属性

number

必須の整数。セッション・ウィンドウ内のフィールド数。

optional

オプションのブール (デフォルトは false)。 60 ページの『オプショナル』 を参照してください。

invertmatch

オプションのブール (デフォルトは false)。 60 ページの『逆ディスクリプ ター』を参照。

hostid

```
オプションのストリング。 67 ページの『セッションの指定』を参照。
```

XML サンプル

<numfields number="10" optional="false" invertmatch="false" />

図 109. <numfields> エレメントの例

<numinputfields> エレメント

概要

<numinputfields> エレメントは、セッション・ウィンドウ内に存在する 3270 また は 5250 入力フィールドの数を指定するディスクリプターです (67 ページの 『「Number of Input Fields」ディスクリプター (<numinputfields> エレメン ト)』を参照)。

属性

number

必須の整数。セッション・ウィンドウ内のフィールド数。

optional

オプションのブール (デフォルトは false)。 60 ページの『オプショナル』 を参照してください。

invertmatch

オプションのブール (デフォルトは false)。 60 ページの『逆ディスクリプ ター』を参照。

hostid

オプションのストリング。 68 ページの『セッションの指定』を参照。

XML サンプル

<numinputfields number="10" optional="false" invertmatch="false" />

図 110. <numinputfields> エレメントの例

<oia> エレメント

概要

<oia> エレメントは、セッション・ウィンドウ内の入力禁止標識の状態を記述する ディスクリプターです (65 ページの『「OIA が禁止解除になるのを待つ (Wait for OIA to Become Uninhibited)」ディスクリプター (<oia> エレメント)』を参 照)。

属性

status 必須。次の値にすることができます。

NOTINHIBITED

マクロ・ランタイムは、入力禁止標識がクリアされているときはこのディスクリプターを true として評価し、入力禁止標識が設定されている場合は false として評価します。

• DONTCARE

マクロ・ランタイムは、常にこのディスクリプターを true として評価します。

• NOTINHIBITED か DONTCARE のどちらかに評価される式

マクロ・ランタイムは、この式を評価してから、その結果に応じて、通 常通りにディスクリプターを評価します。

optional

オプションのブール (デフォルトは false)。 60 ページの『オプショナル』 を参照してください。

invertmatch

オプションのブール。 60 ページの『逆ディスクリプター』を参照。

hostid

オプションのストリング。 66 ページの『セッションの指定』を参照。

XML サンプル

<oia status="NOTINHIBITED" optional="false" invertmatch="false" />

図 111. <oia> エレメントの例

<pause> エレメント

概要

<pause> エレメントは、指定されたミリ秒数の間、待機します (107 ページの『休 止アクション (<pause> エレメント)』を参照)。

属性

value オプションの整数。待機するミリ秒数。この属性を指定しない場合、マクロ・オブジェクトは、スクリプトを保管するときに、属性 "value=10000"
 (10 秒)をこのエレメントに追加します。

ignorepauseoverride

オプションのブール (デフォルトは false)。3270 ディスプレイ・セッション

専用。この属性を true に設定すると、マクロ・ランタイムは、 <HAScript> エレメントの ignorepauseforenhancedtn 属性が true に設定 されている場合であっても、<pause> エレメントを処理します (150 ペー ジの『画面の完了に関係する属性』を参照)。

XML サンプル

<pause timeout="5000">

図 112. <pause> エレメントの例

<perform> エレメント

概要

<perform> エレメントは、インポートした Java クラスに属するメソッドを呼び出します (158 ページの『Java クラスのインポート型の作成』を参照)。

<perform> エレメント以外の多くのコンテキストでも、メソッドを呼び出すことが できます。しかし、値を戻さないメソッドを呼び出したい場合は、<perform> エレ メントが便利です (107 ページの『実行アクション (<perform> エレメント)』を 参照)。

属性

value 必須。変数と同じように、メソッド呼び出しをドル記号 (\$) で囲む必要があ ります (172 ページの『メソッド呼び出しの構文』を参照)。マクロ・エデ ィターで実行アクションを作成する場合に使用するのと同じ形式で、メソッ ド呼び出しのパラメーター (ある場合) を指定する必要があります。

XML サンプル

<!-- Call the update() method associated with the class to which importedVar belongs (such as mypackage.MyClass). --> <perform value="\$importedVar.update(5, 'Application', \$str\$)\$" />

図 113. <perform> エレメントの例

<playmacro> エレメント

概要

<playmacro> エレメントは、現行のマクロを終了し、別のマクロを起動します (109 ページの『PlayMacro アクション (<playmacro> エレメント)』を参照)。こ のプロセスは、マクロのチェーニングと呼ばれます。

<playmacro> エレメントを <actions> エレメント内に配置する場所には、制限があ ります (109 ページの『PlayMacro アクションの追加』を参照)。

Host Access Toolkit を使用する場合は、次のアクションを実行する必要があります。

- MacroManager bean を使用するか、独自の MacroIOProvider クラスをインプ リメントする。管理下のマクロだけをチェーニングできます。
- マクロに名前を割り当てる。マクロは、マクロ名でチェーニングされます。マクロ名は大/小文字が区別されます。

属性

name 必須。ターゲット・マクロの名前。ターゲット・マクロは、呼び出し側マク ロと同じロケーションに存在する必要があります (109 ページの『ターゲ ット・マクロのファイル名と開始画面』を参照)。マクロ名は大/小文字が区 別されます。

startscreen

オプション。マクロ・ランタイムがターゲット・マクロの処理を開始するマ クロ画面 (<screen> エレメント)の名前。マクロ・ランタイムに、ターゲッ ト・マクロの通常の開始画面から開始させるには、値 *DEFAULT* を使用 するか、このパラメーターを省略してください。

transfervars

必須。この属性を Transfer に設定すると、マクロ・ランタイムは、呼び出 し側マクロに属する変数をターゲット・マクロに転送します (110 ページ の『変数の転送』を参照)。デフォルトは No Transfer です。

XML サンプル

図 114. <playmacro> エレメントの例

<print> エレメント

概要

<print> エレメントは、印刷機能を提供します。 3 つの基本印刷アクション (開始、抽出、および終了) は、action 属性を使用して指定されます (111 ページの 『印刷アクション (<print> エレメント)』を参照)。

属性

action 必須。実行される印刷アクション。start、extract、end のいずれかでなけ ればなりません。

start:

マクロ・ランタイムは、ユーザーが指定する「プリンター・セットアッ プ (Printer Setup)」オプションと「ページ・セットアップ (Page Setup)」オプションを使用して、現行のマクロに対して印刷 Bean オブ ジェクトのインスタンスを生成します (112 ページの『印刷開始』を参 照)。

多数のプリンター・セットアップ・オプションとページ・セットアッ プ・オプションがあるので、また、1 つのオプションを変更するにはそ の他の複数のオプションを調整する必要があるので、プリンター・セッ トアップ・オプションとページ・セットアップ・オプションの指定に は、マクロ言語を使用しないでください。代わりに、マクロ・エディタ ーを使用して、印刷開始アクションを作成し、プリンター・セットアッ プ・ウィンドウとページ・セットアップ・ウィンドウを使用して、これ らのオプションを指定してください(112ページの『プリンター・セッ トアップとページ・セットアップ』を参照)。

• extract:

マクロ・ランタイムは、ユーザーが指定するセッション・ウィンドウの 長方形域からテキストをコピーし、そのテキストを現行の印刷 bean に 送信します (113 ページの『印刷抽出』を参照)。

• end:

マクロ・ランタイムは、印刷 bean が存在する場合、その bean を終了 します (114 ページの『印刷終了』を参照)。

- **srow action** が extract である場合は、必須の整数。印刷される長方形のテキス ト域の、行と列の座標の最初のペアの行。
- scol action が extract である場合は、必須の整数。印刷される長方形のテキス ト域の、行と列の座標の最初のペアの列。
- **erow action** が extract である場合は、必須の整数。印刷される長方形のテキス ト域の、行と列の座標の 2 番目のペアの行。
- ecol action が extract である場合は、必須の整数。印刷される長方形のテキス ト域の、行と列の座標の 2 番目のペアの列。

assigntovar

オプションの変数。印刷アクションからの戻りコードを含む変数の名前を指 定します。

hostid

オプションのストリング。この属性は、action が extract の場合のみ使用 できます。 114 ページの『セッションの指定』を参照してください。

XML サンプル

```
<print action="start" assigntovar="$intReturnCode$" />
<print action="extract" srow="1" scol="1" erow="-1" ecol="-1" />
<print action="end" />
```

図 115. <print> エレメントの例

<prompt> エレメント

概要

<prompt> エレメントは、ユーザーに入力を求めるポップアップ・ウィンドウを表示し、ユーザーが OK をクリックするのを待ってから、その入力をセッション・ウィンドウに送信します (114 ページの『プロンプト・アクション (<prompt> エレメント)』を参照)。

属性

name オプションのストリング。'Enter your response here:' のような、ポップ アップ・ウィンドウに表示されるテキスト (115 ページの『プロンプト・ ウィンドウの各部分』を参照)。

description

- オプションのストリング。このアクションの記述。この記述は表示されません (115 ページの『プロンプト・ウィンドウの各部分』を参照)。
- row 必須の整数。マクロ・ランタイムがユーザーからの入力を開始する、セッション・ウィンドウ上の行。
- col 必須の整数。マクロ・ランタイムがユーザーからの入力を開始する、セッシ ョン・ウィンドウ上の列 (117 ページの『セッション・ウィンドウでの入 力シーケンスの処理』を参照)。
- len 必須の整数。ユーザーがプロンプト入力フィールドに入力できる文字数 (117 ページの『応答の長さ』を参照)。

default

オプションのストリング。ポップアップ・ウィンドウの入力フィールドに表 示されるテキスト。ユーザーが入力フィールドに入力するのでなく、単に OK をクリックするだけの場合、マクロ・ランタイムは、このデフォルト入 力をセッション・ウィンドウに送信します (115 ページの『デフォルト応 答』を参照)。

clearfield

オプションのブール。この属性を true に設定すると、マクロ・ランタイム は、入力シーケンスをセッション・ウィンドウに送信する前に、行と列の位 置が存在するセッション・ウィンドウの入力フィールドをクリアします (117 ページの『セッション・ウィンドウでの入力シーケンスの処理』を参 照)。

encrypted

オプションのブール。この属性を true に設定すると、ユーザーがウィンド ウの入力フィールドにキーを入力するときに、マクロ・ランタイムは、その キーに関連した文字ではなく、アスタリスク (*) を表示します (116 ペー ジの『パスワード応答』を参照)。

movecursor

オプションのブール。この属性を true に設定すると、マクロ・ランタイム は、カーソルを入力の末尾に移動します (117 ページの『セッション・ウ ィンドウでの入力シーケンスの処理』を参照)。

xlatehostkeys

オプションのブール。この属性を true に設定すると、マクロ・ランタイム は、アクション・キー (例えば、[enter])の名前を、文字シーケンスとして ではなく、アクション・キーとして解釈します (117 ページの『アクショ ン・キーとホスト・アクション・キーの変換』を参照)。

assigntovar

オプションの変数名。この属性を変数名に設定すると、マクロ・ランタイム は、ユーザーがここで指定する名前を持つ変数に、入力を保管します (118 ページの『変数への入力シーケンスの割り当て』を参照)。

varupdateonly

オプションのブール。この属性を true に設定すると、マクロ・ランタイム は、変数に入力を保管し、その入力をセッション・ウィンドウに送信しませ ん (117 ページの『セッション・ウィンドウでの入力シーケンスの処理』 を参照)。この属性が有効なのは、assigntovar 属性が true に設定されてい る場合だけです。

required

オプションのブール。この属性を true に設定すると、マクロ・ランタイム はプロンプト・ウィンドウの入力フィールドにテキストが格納されないかぎ り、プロンプト・ウィンドウの「OK」ボタンを使用不可にします。入力フ ィールドにテキストが格納されるのは、「デフォルト応答 (Default Response)」が指定されている場合、または入力フィールドにテキストが入 力された場合です (116 ページの『応答の要求』を参照)。

hostid

オプションのストリング。119ページの『セッションの指定』を参照。

XML サンプル

<prompt name="'ID'" row="1" col="1" len="8" description="'ID for Logon'" default="'guest'" clearfield="true" encrypted="true" assigntovar="\$userID\$" varupdateonly="true" required="true"/>

図 116. <prompt> エレメントの例

<recolimit> エレメント

概要

<recolimit> エレメントは、<description>、<actions>、および <nextscreens> エレ メントと同じレベルで、<screen> エレメント内に存在するオプションのエレメント です (143 ページの『認識限度 (「画面 (Screens)」タブの「一般 (General)」タ ブ)』を参照)。

<recolimit> エレメントを使用すると、指定された回数より多くこのエレメントが表示されるマクロ画面をマクロ・ランタイムが処理する場合、処置を取ることができます。

属性

- value 必須の整数。認識限界。マクロ・ランタイムがマクロ画面を何度も認識する 場合、マクロ・ランタイムは、このマクロ画面のアクションを処理するので はなく、指定されたアクションを実行します。
- goto オプションのストリング (デフォルトは、マクロ・ランタイムがエラー・メ ッセージを表示し、マクロを終了することです)。認識限界に達したときに マクロ・ランタイムが処理を開始する、マクロ画面の名前。

XML サンプル

<recolimit value="1" goto="RecoveryScreen1" />

図 117. <recolimit の例

<runprogram> エレメント

概要

<runprogram> エレメントは、ネイティブ・アプリケーションを起動し、オプショ ンとして、そのアプリケーションの終了を待機します。このアプリケーションの入 カパラメーターを指定し、戻りコードを変数に保管することができます (119 ペー ジの『プログラム実行アクション (<runprogram> エレメント)』を参照)。 属性

exe 必須のストリング。ネイティブ・アプリケーションのパスと名前 (119 ペ ージの『ネイティブ・アプリケーションの起動』を参照)。

param

- オプションのストリング。ネイティブ・アプリケーションが起動されるとき に指定される引数。
- wait オプションのブール。この属性を true に設定すると、マクロ・ランタイム は、ネイティブ・アプリケーションが終了するまで待機します。

assignexitvalue

オプションの変数。ネイティブ・アプリケーションからの戻り値が保管され る変数の名前。

XML サンプル

```
<runprogram exe=

"'%ProgramFiles%¥Windows NT¥Accessories¥wordpad.exe'"

param="'c:¥¥tm¥¥new_file.doc'" wait="true"

assignexitvalue="$intReturn$" />

<message title="" value="'Return value is '+

$intReturn$" />
```

図 118. <runprogram> エレメントの例

<screen> エレメント

概要

<screen> エレメント、<import> エレメント、および <vars> エレメントは、<HAScript> エレメント内に存在する 3 つの基本構造エレメントです (20 ページの『マクロ・スクリプトの概念視点』を参照)。

マクロ内には、複数の screen エレメントが存在できます。 1 つの <screen> エレ メントには、1 つのマクロ画面のすべての情報が入っています (24 ページの『マ クロ画面とそのサブコンポーネント』を参照)。

<screen> エレメントには、3 つの基本構造エレメント (<actions> エレメント、
<description> エレメント、および <nextscreens> エレメント) が含まれます (26
ページの『マクロ画面の概念視点』を参照)。

属性

name 必須。この <screen> エレメント (マクロ画面) の名前。この名前は、すで に存在している <screen> エレメントの名前と同じあってはなりません。

entryscreen

オプションのブール (デフォルトは false)。この属性を true に設定する

と、マクロ・ランタイムは、この <screen> エレメントをマクロの有効な開 始画面として取り扱います (139 ページの『入り口画面』を参照)。

exitscreen

オプションのブール (デフォルトは false)。この属性を true に設定する と、マクロ・ランタイムは、この <screen> エレメントをマクロの有効な出 口画面として取り扱います (140 ページの『出口画面』を参照)。

transient

オプションのブール (デフォルトは false)。この属性を true に設定する と、マクロ・ランタイムは、いつでも表示され、常にクリアする必要がある 画面として、この <screen> エレメントを取り扱います (140 ページの 『一時画面』を参照)。

pause オプションの整数 (デフォルトは -1)。この属性の値をミリ秒で指定する と、この <screen> エレメントのマクロ・ランタイムは (<HAScript> エレ メントの pausetime 属性を使用して設定された) "アクション間の休止" に 関するデフォルト遅延を無視し、代わりにこの値を使用します (146 ペー ジの『休止時間の設定 (「画面 (Screens)」タブの「一般」タブ)』を参照)。

XML サンプル

図 119. <screen> エレメントの例

<sqlquery> エレメント

概要

<sqlquery> エレメントはデータベースに SQL ステートメントを送信し、SQL ス テートメントから作成されたデータがあればこれを検索して、グローバル変数に格 納したり、ファイルに書き込んだり、データを表示します (121 ページの 『SQLQuery アクション (<sqlquery> エレメント)』を参照)。

属性

url 必須のストリング。jdbc:as400://myISeries など、SQL ステートメントの 送信先となるデータベース・サーバーのデータベース URL です (123 ペ ージの『データベースの URL』を参照)。

driver 必須のストリング。データベース・サーバーとの接続に使用されるドライバ

ーの完全修飾パッケージ名です (COM.ibm.db2.jdbc.app.DB2DRIVER など)。 このパッケージは、クライアント・ワークステーション上に存在している必 要があります (124 ページの『ドライバー ID とドライバー・クラス』を 参照)。

userid

オプションのストリング。データベースにアクセスするためのユーザー ID (必要な場合) (125 ページの『ユーザー ID とパスワード』を参照)。

password

オプションのストリング。データベースにアクセスするためのパスワード (必要な場合) (125 ページの『ユーザー ID とパスワード』を参照)。

statement

必須のストリング。SQL ステートメントです (125 ページの『ステートメ ント』を参照)。

outputtype

必須の整数。SQL ステートメントから作成されたデータの転送先です。有 効な値は次のとおりです。

- 0 データはグローバル変数 \$HMLSQLUtil\$ に格納されます。データを 検索するには、この変数に関するメソッドを呼び出します(127 ページ の『グローバル変数 (\$HMLSQLUtil\$) へのデータの格納』を参照)。
- 1 データはファイルに書き込まれます(127 ページの『ファイルへの データの書き込み』を参照)。この出力タイプを使用する場合は、以下の 属性も指定する必要があります。
 - outfilename
 - outfiletype
 - overwrite
 - inbrowser
- 2 データはワークステーション・モニターに表示されます(128 ページの『データの表示』を参照)。この出力タイプを使用する場合は、以下の属性も指定する必要があります。
 - holdondisplay

outfilename

ストリング (outputtype が 1 の場合に必須)。出力ファイルの完全パスお よび名前です (127 ページの『ファイルへのデータの書き込み』を参照)。

outfiletype

整数 (outputtype が 1 の場合に必須)。出力ファイルのタイプです (127 ページの『ファイルへのデータの書き込み』を参照)。有効な値は次のとおりです。

- 0 ASCII テキスト (*.txt)
- 1 コンマ区切り値 (*.csv)
- 2 Lotus 1-2-3 (*.wk1)
- 3 Microsoft Excel BIFF3 (*.xls)
- 4 Microsoft Excel BIFF4 (*.xls)
- 5 XML (*.xml)

• 6 - HTML (*.html)

overwrite

ブール (outputtype が 1 の場合に必須)。ファイルを上書きするか、ファ イルに追加するかのいずれかを指定します (127 ページの『ファイルへの データの書き込み』を参照)。

- この属性を true に設定すると、指定したファイルが存在する場合、この ファイルがデータで上書きされます。指定したファイルが存在しない場 合は、作成されます。
- この属性を false に設定すると、指定したファイルが存在する場合、このファイルにデータが追加されます。指定したファイルが存在しない場合は、作成されます。

inbrowser

ブール (outputtype が 1 の場合に必須)。この属性を true に設定すると、 マクロ・ランタイムは指定したファイルにデータを書き込んだ後に、ファイ ルの内容をデフォルト・ブラウザーに表示します (127 ページの『ファイ ルへのデータの書き込み』を参照)。

holdondisplay

ブール (outputtype が 2 の場合に必須)。この属性を true に設定すると、 マクロ・ランタイムはデータ表示後にエンド・ユーザーからの応答があるま で待機してから、次のマクロ・アクションの処理を開始します (128 ペー ジの『データの表示』を参照)。

mlprops

オプションのストリング。outfiletype が 5 (XML) または 6 (HTML) の場 合、この属性にはファイル設定が格納されます。 この属性のフォーマット は、次のとおりです。

mlprops="*key1*0@*value1* @@*key2*0@*value2*0@*key3*0@*value3*"

ここで、key1、key2、key3 などは HTML または XML 設定の名前です。 value1、value2、value3 などは HTML または XML 設定の値です。コー ド・エディターを使用してこの属性を手動で設定できますが、これらの設定 を変更する場合は、 SQL Wizard の対応する設定を変更し、値をマクロ・ エディターに保管することを推奨します (121 ページの『SQL ウィザード の使用』を参照)。

XML サンプル

```
<sqlquery url="'jdbc:as400://elcrtp06'"
    driver="'com.ibm.as400.access.AS400JDBCDriver'"
    userid="'myuser'"
    password="Ex0bRtrf73mPrwGrWMT+/g=="
    statement="'SELECT * FROM HODTEST WHERE ((HODTEST.DESCRIPT is not null))'"
    outputtype="1"
    outfilename="'MyFile'"
    outfiletype="4"
    overwrite="true"
    inbrowser="false" />
```

図 120. <sqlquery> エレメントの例

<string> エレメント

概要

<string> エレメントは、文字のシーケンス、およびそのシーケンスがあるセッショ ン・ウィンドウの長方形域を指定するディスクリプターです (69 ページの『スト リング・ディスクリプター (<string> エレメント)』を参照)。

文字のシーケンスは、長方形ブロック内の任意の場所に存在することができます。

属性

value 必須のストリング。文字のシーケンス。

- row オプションの整数 (デフォルトは画面全体の検索)。長方形のテキスト・ブロ ックの一方のコーナーの行位置。
- col オプションの整数。長方形のテキスト・ブロックの一方のコーナーの列位 置。
- erow オプションの整数。長方形のテキスト・ブロックの向かい側のコーナーの行 位置。
- ecol オプションの整数。長方形のテキスト・ブロックの向かい側のコーナーの列 位置。

casesense

オプションのブール (デフォルトは false)。この属性を true に設定する と、マクロ・ランタイムは、大/小文字の区別のあるストリング比較を実行 します。

- wrap オプションのブール (デフォルトは false)。
 - この属性を false に設定すると、マクロ・ランタイムは、長方形のテキ スト・ブロックの各行で文字のシーケンスを検索します。文字のシーケ ンスが次の行に折り返す場合、マクロ・ランタイムはその折り返しを検 出しません。
 - この属性を true に設定すると、マクロ・ランタイムは、任意の行にある 文字のシーケンス、または長方形のテキスト・ブロックの次の行に折り

返す文字のシーケンスがないかどうかを調べます (70 ページの『マク ロ・ランタイムが長方形域を検索する方法 (折り返しオプション)』を参 照)。

optional

オプションのブール (デフォルトは false)。 60 ページの『オプショナル』 を参照してください。

invertmatch

オプションのブール。 60 ページの『逆ディスクリプター』を参照。

hostid

オプションのストリング。 72 ページの『セッションの指定』を参照。

XML サンプル

<!-- The string must occur in one specific area of a single row -->
<string value="'Utility Selection Panel'" row="3" col="28"
 erow="3" ecol="51" casesense="false" wrap="false"
 optional="false" invertmatch="false" />
<!-- The string can occur in any single row of the session area -->

<string value="'Utility Selection Panel'" row="1" col="1"
 erow="-1" ecol="-1" casesense="false" wrap="false"
 optional="false" invertmatch="false" />

図 121. <string> エレメントの例

<trace> エレメント

概要

<trace> エレメントは、ユーザーが指定するトレース宛先 (例えば、Java コンソー ル) に、トレース・メッセージを送信します (129 ページの『トレース・アクショ ン (<trace> エレメント)』を参照)。

属性

- **type** 必須。トレース・データの宛先。この宛先は、次のいずれかでなければなり ません。
 - HODTRACE: Host On-Demand トレース機能
 - USER: ユーザー・トレース・ハンドラー
 - SYSOUT: Java コンソール

value 必須のストリング。トレースの宛先に送信されるストリング。

XML サンプル

<trace type="SYSOUT" value="'The value is '+\$strData\$" />

図 122. <trace> エレメントの例

<type> エレメント

概要

<type> エレメントは、Java クラス (例えば、java.util.Properties) を表すインポ ート型 (例えば、Properties) を宣言します。型を宣言した後、その型に基づいて変 数を作成し、Java クラスのインスタンスを作成し、そのインスタンスでメソッドを 呼び出すことができます (158 ページの『Java クラスのインポート型の作成』を参 照)。

また、型は、静的メソッドを直接呼び出すのにも使用できます (インスタンスを生 成する必要がありません)。

<type> エレメントは、<import> エレメント内に存在する必要があります。

属性

- **class** 必須。パッケージ名 (例えば、java.util.Properties) があれば、これを含 めて、インポートされるクラスの完全修飾クラス名。
- name オプション。インポート型を参照するためにマクロ内の他の場所で使用でき る、短い名前 (例えば、Properties)。短い名前を指定しない場合、短い名前 は、完全修飾クラス名と同じになります。型名のスペルには、いくつかの制 限があります (160 ページの『変数名と型名』を参照)。

XML サンプル

<import> <type class="i

```
<type class="java.util.Date" name="Date"/>
<type class="java.io.FileInputStream"/>
<type class="com.ibm.eNetwork.beans.HOD.HODBean" name="HODBean"/>
<type class="myPackage.MyClass" name="MyClass"/>
</import>
```

<vars> エレメント

概要

<vars> エレメント、<import> エレメント、および <screen> エレメントは、<HAScript> エレメント内に存在する 3 つの基本構造エレメントです (20 ページの『マクロ・スクリプトの概念視点』を参照)。

<vars> エレメントはオプションです。このエレメントには、複数の <create> エレ メントが含まれ、各エレメントは、変数を宣言し、初期化します (157 ページの 『変数の新規作成』を参照)。 <vars> エレメントは、<import> エレメントの後、 かつ最初の <screen> エレメントの前に存在しなければなりません。

変数を使用するには、<HAScript> 内の usevars エレメントを true に設定する必 要があります。

属性

なし。

XML サンプル

```
<HAScript ... usevars="true" .... >
    <import>
        <type class="java.util.Properties" name="Properties" />
        </import>
        <vars>
            <create name="$prp$" type="Properties" value="$new Properties()$" />
            <create name="$strAccountName$" type="string" value="" />
            <create name="$intAmount$" type="integer" value="0" />
            <create name="$intAmount$" type="double" value="0.0" />
            <create name="$bolSignedUp$" type="boolean" value="false" />
            <create name="$fldFunction$" type="field" />
            </vars>
            ...
</HAScript>
```

図 123. <vars> エレメントの例

<varupdate> エレメント

概要

<varupdate> エレメントにより、マクロ・ランタイムは、指定された値を指定され た変数に保管します。この値は、即時値、変数、Java メソッドの呼び出し、または これらの値のいずれかを含むことができる演算式にすることができます。値が式で ある場合、マクロの再生時に、マクロ・ランタイムはその式を評価し、指定された 変数に結果の値を保管します (131 ページの『変数更新アクション (<varupdate> エレメント)』を参照)。

また、<description> エレメント内の <varupdate> アクションを使用することもで きます (131 ページの『変数更新アクション (<varupdate> エレメント)』を参 照)。

変数の詳細については、 153 ページの『第 11 章 変数とインポートした Java ク ラス』を参照してください。

属性

name 必須。変数の名前。

value 必須のストリング。変数に割り当てられる値または式。

XML サンプル

```
<type>
   <type class="mypackage.MyClass" name="MyClass" />
   <type class="java.util.Hashtable" name="Hashtable" />
   <type class="java.lang.Object" name="Object" />
</type>
<vars>
   . . .
</vars>
<screen>
   <description>
   </description>
   <actions>
      <varupdate name="$var_boolean1$" value="false" />
      <varupdate name="$var_int1$"
                                         value="5" />
      <varupdate name="$var_double1$" value="5" />
      <varupdate name="$var_string1$" value="'oak tree'" />
<varupdate name="$var_field1$" value="4,5" />
      <!-- null keyword -->
      <varupdate name="$var_importedMC1$" value="null" />
      <!-- Equivalent to null keyword for an imported type -->
      <varupdate name="$var_importedMC2$" value="" />
      <varupdate name="$var importedMC4$"
                  value="$new MyClass( 'myparam1', 'myparam2' )$" />
      <varupdate name="$var importedMC5$"</pre>
                  value="$var_importedMC4$" />
      <varupdate name="$var_importedMC6$"
                  value="$MyClass.createInstance( 'mystringparam1' )$" />
      <varupdate name="$var boolean2$"
                  value="$var_importedMC4.isEmpty()$" />
      <varupdate name="$var int2$"
                  value="$($var importedMC4.getHashtable()$).size()$" />
      <varupdate name="$var double2$"
                  value="$var_importedMC4.getMeters()$" />
      <varupdate name="$var string2$"</pre>
                  value="$var importedMC4.toString()" />
   </actions>
</screen>
```

図 124. <varupdate> エレメントの例

第17章 サンプル・マクロ・コード

Excel スプレッドシートまたは DB2 データベースへの CICS トランザク ション・レコードのコピー

概要

注: このサンプルには、Microsoft Excel または IBM DB2 が必要です。

このサンプル・マクロは、非常に小さいサンプル・データベースである CICS トラ ンザクション amnu から、トランザクション・レコードを読み取り、それらのレコ ードを Excel スプレッドシートまたは IBM DB2 に書き込みます。

このサンプル用のファイルは、下記のディレクトリーに保管されます。ここで、 <install> は Host On-Demand インストール・ディレクトリーを表し、xx はご使 用の言語 ID (例えば、en) を表します。

<install>#HOD#xx#doc#macro#samples#amnu

このサンプル内のファイルは次のとおりです。

• amnu.mac、amnudb2.mac

これらはマクロ・ファイルです。最初のファイルは、Excel スプレッドシート用 です。 2 番目のファイルは、DB2 用です。

• amnu.xls

これは、Excel スプレッドシートです。

• EditDB.java、EditDB.jar

EditDB.java は、EditDB クラス用のソース・コードが入っている Java ソース・ ファイルです。マクロは EditDB クラスを使用して、スプレッドシートまたはデ ータベースに書き込みます。 EditDB.class は Java 2 でコンパイルされ、 EditDB.jar に保管されます。 EditDB.jar は Java 2 JAR ファイルです。

Excel サンプルの実行手順 (Windows のみ)

1. Java セキュリティー・ポリシーを構成する。

このサンプルを実行するには、Host On-Demand アプレットに対する所定の許 可を付与する必要があります。 policytool を使用して .java.policy ファイルを 変更するか、新しいポリシー・ファイルを作成し、プラグイン Java ランタイ ム・パラメーター (-Djava.security.policy=PolicyFileName) でこのファイル を指定することができます。

新しいポリシー・ファイルは、次のものを含む必要があり、ローカル・ホーム・ ディレクトリーに置かれなければなりません。

```
grant {
```

```
permission java.lang.RuntimePermission
    "accessClassInPackage.sun.jdbc.odbc";
permission java.util.PropertyPermission
    "file.encoding", "read"; };
```

.java.policy を変更する (プラグインで上記のパラメーターを設定しない) 場 合、Java プラグイン・インストールの bin ディレクトリーで policytool 実行 プログラムを起動し、上記の行で指定された許可を設定してください。

- 2. Excel スプレッドシートを ODBC データ・ソースとしてセットアップする。
 - a. Windows マシンで、「設定」->「コントロール パネル」->「管理ツー ル」->「データ ソース (ODBC)」の順に進む。
 - b. 「追加」をクリックする。
 - c. Microsoft Excel Driver (*.xls) を選択する。
 - d. 「完了」をクリックする。
 - e. データ・ソース名 amnu を入力し、必要な説明を入力する (またはブラン クのままにする)。
 - f. 「ブックの選択」ボタンを使用して、この例で提供されるスプレッドシート を見付ける。OK をクリックします。
 - g. このソースの「読み取り専用」オプションの選択を解除する。このボタンを 見付けるには、「オプション>>」ボタンのクリックが必要な場合がありま す。
 - h. OK をクリックする。これで、amnu.xls スプレッドシートが ODBC デー タ・ソース amnu として使用可能になりました。
- Host On-Demand クライアントが EditDB クラスにアクセスできるようにす る新しい「デプロイメント・ウィザード (Deployment Wizard)」ページを作成 する。
 - a. デプロイメント・ウィザードを開始する。
 - b. 「追加オプション (Additional Options)」ページで、「拡張オプション (Advanced Options...)」をクリックする。
 - c. 「HTML パラメーターの追加 (Add HTML Parameters)」パネルで、名前 「AdditionalArchives」と値「amnu」を持つパラメーターを追加する。
- 4. amnu.jar を Host On-Demand パブリッシュ・ディレクトリーに置く。
- 5. 新たに作成されたページを Web ブラウザーで開き、CICS セッションを開始す る。

AMNU の使用:

トランザクション amnu は、CICS に付属の小さいサンプル・データベースで す。amnu を開始する手順は、次のとおりです。

- a. CICS にログオンする
- b. CICS プロンプトで、amnu と入力し、Enter を押す。

セッション・ウィンドウの左上の四分区間に amnu メニューが表示され、演算 子命令を表示します。

データベースにレコードがあるかどうかを調べる手順は、次のとおりです。

- a. ENTER TRANSACTION: フィールドに、abrw と入力する。
- b. NUMBER フィールドをブランクのままにする。
- c. Enter を押す。

amnu トランザクションは最初の 4 つのレコードを表示します。画面上の指示 にしたがって、データベースをブラウズします。

データベースが空である場合、マクロを実行する前にレコードを追加する必要が あります。データベースにレコードを追加する手順は、次のとおりです。

- a. ENTER TRANSACTION: フィールドに、aadd と入力する。
- b. NUMBER フィールドに、追加したいレコードの番号 (例えば、40) を入力 する。
- c. Enter を押す。
- d. 画面上の指示にしたがって、新しいレコードに情報を指定する。
- 6. amnu.mac をセッションにロードする。
 - a. マクロ・マネージャー・ツールバーが表示されていない場合は、セッショ ン・ツールバーで「表示 (View)->「マクロ・マネージャー」の順に選択し て、マクロ・マネージャー・ツールバーを表示する。
 - b. 「現行マクロのプロパティーを編集 (Edit current macro properties)」アイ コンをクリックする。
 - c. マクロ・エディターで「インポート (Import...)」ボタンをクリックする。
 - d. amnu.mac のロケーションまでブラウズし、amnu.mac を開く。
 - e. 「保管して終了」をクリックして、現行セッションにマクロを保管し、マク ロ・エディターを閉じる。
- 7. amnu メニュー画面にナビゲートし、「マクロを再生 (Play macro)」アイコン をクリックする。

レコード番号を入力するように求められます。開始したトランザクションの番 号、またはステップ 5 でデータベースをブラウズしたときに表示されたトラン ザクションの番号を入力します。 OK をクリックしてください。入力した番号 に対応するレコードの内容が、アプリケーション画面に表示されます。画面上の レコードをデータベースに保管したいかどうかをたずねるプロンプトが、表示さ れます。デフォルトの応答は「Y」です。OK をクリックしてください。トラン ザクション番号を入力するように求められます。必要な数のレコード番号を続け て入力できます (無効な番号を入力した場合は通知されます) が、終了するに は、Q をクリックします。 OK をクリックしてプロンプトを閉じ、もう一度 OK をクリックして、メッセージ「Good Bye!」を消します。マクロが終了し、 amnu.xls が開きます。スプレッドシート内に、保管したばかりのレコードの内 容が表示されます。

うまくいけば、このサンプルにより、ビジネスに使用するマクロを書き込む強力 な方法を思い付くことができます。このサンプルは簡単に変更できることに注目 してください。例えば、別の種類のデータベースに書き込んだり (DB2 に書き 込む方法については、下記を参照)、ローカル・データベースから読み取って、 amnu データベースに書き込んだりすることができます。このサンプルの目的 は、Java やマクロのコーディングの最良な実例のレッスンではなく、短く簡単 に設計することです。例えば、レコードを書き出すたびに、ローカル・データベ ースに接続したり、切断したりしていることに気付きます。これを避けるには、 マクロの再生ごとに 1 回の接続と 1 回の切断だけがあるように、amnu マク ロにリンクされる「connect」マクロと「disconnect」マクロを作成できます。

Excel マクロ amnu.mac を調べると、Excel スプレッドシートとの接続に sun.jdbc.odbc.JdbcOdbcDriver ドライバーを使用していることが分かります。この クラスがご使用のクラスパス内にないと、このサンプルは正しく実行されません。 このクラスは、IBM プラグインではなく、Sun Java 2 プラグインに含まれていま す。

DB2 サンプルの実行手順

IBM プラグインを使用しているときに、クラスパス内に sun.jdbc.odbc package が ない場合、このサンプルを IBM DB2 で実行できます。

- DB2 データベースを作成する。そのデータベースに「AMNU」という名前を付け、列 「NAME」、「ADDRESS」、「PHONE」、「DATE」、「AMOUNT」、および「COMMENT」があるテーブル「CUSTRECS」を作成します。
- 該当する DB2 ドライバーをクラスパスに置く。 sun.jdbc.odbc.JdbcOdbcDriver を使用するのではなく、COM.ibm.db2.jdbc.net.DB2Driver を使用して、ローカ ル・データベースに接続します。このクラスとその他の必要なクラスは、 db2java.zip にあります。これは、DB2 をインストールしたときに ¥SQLLIB¥java に置かれているはずです。これらの必要なファイルをクラスパス に入れるには、ご使用のセットアップに応じて複数の方法があります。
- 3. DB2 データベース用にマクロを編集する。
 - a. 次の行で、

```
<create name="$database$" type="DB"
value="$new DB('jdbc:db2://hostname:6789/AMNU',
'COM.ibm.db2.jdbc.net.DB2Driver')$" />
```

- ホスト名を、DB2 を実行しているマシンの名前に変更する。
- b. 次の 2 つの perform アクションを見付ける。

<perform value="\$database.setUserID('db2admin')\$" />
<perform value="\$database.setPassword('db2admin')\$" />

DB2 データベースに接続する ID とパスワードを使用して、これらのアク ションを変更します。

4. 上記のステップ 3 から 7 を実行して、AMNUDB2.mac をインポートする。

この2つのマクロには、相違点がいくつかあることに注意してください。

• SQL クエリーにおけるテーブル名の構文が異なります。 Excel の場合: [CUSTRECS\$]

DB2 の場合:

CUSTRECS

前述のように、データベースとの接続に異なるドライバーを使用します。

- DB2 の場合は、データベースに接続するための適切な ID とパスワードを指定 するのに、EditDB の setUserID と setPassword メソッドを使用する必要があ りました。
- このマクロは、追加が終了すると、ローカル・データベースを起動しません。デ ータベース上で「Select * from CUSTRECS」クエリーを実行すると、レコード が正しく追加されたことを確認できます。

付録 A. 追加情報

1 つのマクロ画面内の複数ディスクリプターのデフォルト結合規則

規則の記述

この規則は次のとおりです。

- 1. すべての必須ディスクリプター (つまり、「オプショナル (Optional)」フィール ドが false に設定されているディスクリプター) を評価する。
 - a. すべて true である場合、画面が一致します。
 - b. それ以外の場合は、ステップ 2 に進みます。
- 2. オプション・ディスクリプター (「オプショナル (Optional)」フィールドが true に設定されているディスクリプター) の評価を開始する。
 - a. いずれかのオプション・ディスクリプターが true である場合、画面が一致 します。
 - b. それ以外の場合は、オプション・ディスクリプターの評価を続けます。
- 3. この段階に達した場合、このマクロ画面はアプリケーション画面と一致しません。

入力アクションの略号キーワード

ここでは、入力アクションの略号キーワード、および略号がサポートされているセッションのタイプを説明します。所定の略号に対するセッション・サポートは X で示され、その機能に適用される特別な注が付きます。

機能:	キーワード:	3270:	5250:	VT:	CICS:
Attention	[attn]	x	x		x
Alternate	[altview]	x ³	x ³		x ³
view					
Backspace	[backspace]	x	x	\mathbf{x}^1	x
Backtab	[backtab]	x	x		x
Beginning of	[bof]	x	x		x
Field					
Clear	[clear]	x	x	\mathbf{x}^1	x
Cursor Down	[down]	x	x	\mathbf{x}^1	x
Cursor Left	[left]	x	x	\mathbf{x}^1	x
Cursor Right	[right]	x	x	\mathbf{x}^1	x
Cursor Select	[cursel]	x	x	\mathbf{x}^1	х
Cursor Up	[up]	x	x	\mathbf{x}^1	x
Delete	[delete]	x	x	x ^{1, 2}	x
Character					

表 35. 入力アクションのキーワード

機能:	キーワード:	3270:	5250:	VT:	CICS:
Display SO/SI	[dspsosi]	x ³	x ³		x ³
Dup Field	[dup]	x	x		x
Enter	[enter]	x	x	x	x
End of Field	[eof]	x	x	x ^{1, 2}	x
Erase EOF	[eraseeof]	x	x		х
Erase Field	[erasefld]	x	x		х
Erase Input	[erinp]	x	x		x
Field Exit	[fldext]		x		
Field Mark	[fieldmark]	x	x		
Field Minus	[field-]		x		
Field Plus	[field+]		x		
F1	[pf1]	x	x	x	x
F2	[pf2]	x	x	x	x
F3	[pf3]	x	x	x	x
F4	[pf4]	x	x	x	x
F5	[pf5]	x	x	x	x
F6	[pf6]	x	x	x	x
F7	[pf7]	x	x	x	x
F8	[pf8]	x	x	x	x
F9	[pf9]	x	x	x	x
F10	[pf10]	x	x	x	x
F11	[pf11]	x	x	x	x
F12	[pf12]	x	x	x	x
F13	[pf13]	x	x	x	x
F14	[pf14]	x	x	x	x
F15	[pf15]	x	x	x	x
F16	[pf16]	x	x	x	x
F17	[pf17]	x	x	x	x
F18	[pf18]	x	x	x	x
F19	[pf19]	x	x	x	x
F20	[pf20]	x	x	x	x
F21	[pf21]	x		x	x
F22	[pf22]	x		x	x
F23	[pf23]	x		x	x
F24	[pf24]	x		x	x
Help	[help]		x		
Home	[home]	x	x	x ^{1, 2}	x
Insert	[insert]	x	x	x ^{1, 2}	x
Keypad 0	[keypad0]			x	

表 35. 入力アクションのキーワード (続き)

機能:	キーワード:	3270:	5250:	VT:	CICS:
Keypad 1	[keypad1]			x	
Keypad 2	[keypad2]			x	
Keypad 3	[keypad3]			x	
Keypad 4	[keypad4]			x	
Keypad 5	[keypad5]			x	
Keypad 6	[keypad6]			x	
Keypad 7	[keypad7]			x	
Keypad 8	[keypad8]			x	
Keypad 9	[keypad9]			x	
Keypad Dot	[keypad.]			x	
Keypad Enter	[keypadenter]			x	
Keypad Comma	[keypad,]			x	
Keypad Minus	[keypad-]			x	
New Line	[newline]	x	x		x
PA1	[pa1]	x	x		x
PA2	[pa2]	x	x		x
PA3	[pa3]	x	x		x
Page Up	[pageup]	x	x	x ^{1, 2}	x
Page Down	[pagedn]	x	x	x ^{1, 2}	х
Reset	[reset]	x	x	x	x
System Request	[sysreq]	x	x		x
Tab Field	[tab]	x	x	x ¹	x
Test Request	[test]		x		

表 35. 入力アクションのキーワード (続き)

- VT はこの機能をサポートしますが、この機能に影響を与えるのは、ホスト・ア プリケーションです。
- 2. VT200 モードのみでサポートされます。
- 3. この機能は、DBCS セッションのみで使用可能です。

次の表は、入力アクションの双方向キーワードを示しています。

表 36. 入力アクションの双方向キーワード

機能:	キーワード:	3270:	5250:	VT:	CICS:
Auto Push	[autopush]	x			x
Auto Reverse	[autorev]	x		х	
Base	[base]	x	x		
BIDI Layer	[bidilayer]				

機能:	キーワード:	3270:	5250:	VT:	CICS:
閉じる	[close]		x		
(Close)					
CSD	[csd]	x			x
End Push	[endpush]	x			х
Field Reverse	[fldrev]	x	x		x
Field Shape	[fieldshape]	x			х
Final	[final]	x			x
Initial	[initial]	x			х
Isolated	[isolated]	x			x
Latin Layer	[latinlayer]	x	x		
Middle	[middle]	x			x
Push	[push]	x			x
Screen	[screenrev]	x	x		x
Reverse					

表 36. 入力アクションの双方向キーワード (続き)

付録 B. 特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合 があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービス に言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能 であることを意味するものではありません。これらに代えて、IBM の知的所有権を 侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用す ることができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの 評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を 保有している場合があります。本書の提供は、お客様にこれらの特許権について実 施権を許諾することを意味するものではありません。実施権についてのお問い合わ せは、書面にて下記宛先にお送りください。

〒103-8510 東京都中央区日本橋箱崎町19番21号 日本アイ・ビー・エム株式会社 法務・知的財産 知的財産権ライセンス渉外

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 IBM お よびその直接または間接の子会社は、本書を特定物として現存するままの状態で提 供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むす べての明示もしくは黙示の保証責任を負わないものとします。国または地域によっ ては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限 を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的 に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随 時、この文書に記載されている製品またはプログラムに対して、改良または変更を 行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜の ため記載しただけであり、決してそれらの Web サイトを推奨するものではありま せん。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではあり ません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプロ グラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の 相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする 方は、下記に連絡してください。 IBM Corporation Department T01 Building B062 P.O. Box 12195 Research Triangle Park, NC 27709-2195 U.S.A.

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができま すが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、 IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれ と同等の条項に基づいて、IBM より提供されます。

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができま すが、有償の場合もあります。本書で説明されているライセンス・プログラムまた はその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログ ラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公 に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っ ておりません。したがって、他社製品に関する実行性、互換性、またはその他の要 求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの 製品の供給者にお願いします。

この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は表示さ れない場合があります。

付録 C. 商標

IBM、IBM ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corp. の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml をご覧ください。

Java およびすべての Java 関連の商標およびロゴは Oracle やその関連会社の米国 およびその他の国における商標または登録商標です。

Microsoft、Windows および Windows ロゴは、Microsoft Corporation の米国お よびその他の国における商標です。


Printed in Japan

SC88-9760-07



日本アイ・ビー・エム株式会社 〒103-8510東京都中央区日本橋箱崎町19-21