



[プライマネージ]

**PRIMANAGE**

CSS Open Package Series for Enterprise Resource Planning

*PRIMANAGE MEISTER*

ソフトウェア部品システムのエラーログ

*Central Soft Service Co.,Ltd.*

株式会社セントラルソフトサービス

## 目 次

付録 A ソフトウェア部品システムのエラーログ .....	3
〔1〕 エラーログの記録されているファイル .....	3
〔2〕 エラーログの内容 .....	4
〔3〕 記録される追加情報と、その解説 .....	6

## 付録 A ソフトウェア部品システムのエラーログ

ソフトウェア部品システムには、実行中にエラーが発生した場合、そのエラーがどのようなものであったのか、を記録して残す機能があります。ソフトウェア部品システムでは、この記録された情報を、エラーログと呼んでいます。

### [1] エラーログの記録されているファイル

- エラーログは、テキストファイルの中に書き込まれています。このファイルは、エディタや Notepad で開いて読むことができます。
- このテキストファイル（以降、ログファイルと呼びます）は、日の単位で作成されます。ある一日に発生した、全てのエラーの内容は、同じログファイルの中に書き込まれ、日付が変わると、新しいログファイルが用意されます。ログは、それまでの記録に追記される形で記録されるので、一日のうちに複数のエラーが発生している場合は、それぞれのログは、ログファイルの先頭から、エラーの発生順に書き込まれています。つまり、一番新しいエラーについてのログは、ログファイルの最後に記録されています。
- ログファイルの名前は、“ope\_log.nnnnnn” という形式になっています。エラーログに特有の拡張子はありません。（強いていえば、“nnnnnn” が拡張子に相当します。）名前の中の“nnnnnn”は、6桁の数字で、ログを記録した年月日を表します。6桁の数字の先頭の2桁は年を表していて、西暦で数えた年の下二桁が割り当てられています。次の二桁が月、最後の二桁が日を表しています。例えば、2006年5月11日に発生したエラーのログは、ope\_log.060511 の中に書き込まれています。
- ログファイルは、環境変数 BSS\_HOME で指定されるディレクトリの直下にある、“errlog” というディレクトリの下に置かれます。（=%BSS\_HOME%¥errlog）

[注] SWB MEISTER がインストールされているシステムでは、errlog ディレクトリの中に、“tool\_log.nnn” という形式のファイルが作られていることがあります。これもエラーのログファイルですが、このファイルは、SWB MEISTER の、DBTool を実行してエラーが発生したときに、DBToolg が作成したものです。“ope\_log.nnnnnn” とは別のものです。

## [2] エラーログの内容

- エラーは、第1相、第2相、第3相、の各相で発生することがありますが、どの相で発生したエラーでも、同一の形式でログファイルに書き込まれます。その形式は、次のようになっています。(nには数字が入り、Xにはアルファベットが入ります。)

エラーコード = nnnn  
エラーID = X  
処理フラグ = nn  
エラーメッセージ：  
追加情報：  
エラー検知プログラム名：  
ホスト名：  
障害記録日付：nn/nn/nn  
障害記録時刻：nn:nn

- 各項には次のような意味があります。

### a) エラーコード

- \* ソフトウェア部品システムで、あらかじめ定義されているエラーの種類を、コードで表したものです。

例えば、

“1219” → “日付の存在エラー(カレンダーにない日付)”

“6100” → “[BSS\_HOME] が環境変数に設定されていない”

“7010” → “データベースオープンエラー”

等を表します。

- \* エラーコードの種類と、それぞれのコードがどのようなエラーを表しているのか、は C\_err.h に記載されています。

### b) エラーID

- \* このIDは、エラーの重大さの程度を表しています。次の、“I”、“W”、“E”、“F”の中のどれかが使われます。

エラーID	重大さの程度
"I"	エラーでなく情報
"W"	警告
"E"	エラー
"F"	システム障害

- \* エラーに対して、その重大がどの程度のものなのか、は 部品を作成する人が判断します。

### c) 処理フラグ

- \* 30 に固定されています。(30 は、エラーログ出力を表します。)

### d) エラーメッセージ

- \* エラーメッセージを書き込む領域ですが、現在はほとんど使われていません。一方、次項の追加情報の項は、よく使われます。

### e) 追加情報

- \* エラーについての、付加情報を書き込む領域です。エラーの原因を探る上で重要な情報が書き込まれていることがあります。この項については、下の“(3) 記録される追加情報と、その解読”をご参照ください。

- f) エラー検知プログラム名
  - \* エラーが発生したときに、実行中だったモジュールの名前が書き込まれます。モジュールの名前ばかりでなく、エラーが発生したときに、実行中だった関数名が書き込まれていることもあります。
- g) ホスト名
  - \* エラーが第 2 相、あるいは第 3 相で発生したときには、サーバー・システムの名前が書き込まれます。第 1 相で発生したときは、クライアント・システムの名前が書き込まれます。
- h) 障害記録日付
  - \* エラーが発生した日付が書き込まれます。(06/05/11 のように、スラッシュで年月日を分離したフォーマットが使われています。)
- i) 障害記録時刻
  - \* エラーが発生した時間が記録されています。(“14:50” のように、時と分が記録され、時と分の間がコロンで分離されている形式をしています。秒は記録されません。)

### [3] 記録される追加情報と、その解読

第 2 相と第 3 相では、エラーが発生したときに、その原因を究明するときの役に立つように、補助となる情報を、追加情報の欄に記録します。

第 2 相と第 3 相では、追加情報に記録する内容と形式が異なりますから、別々に説明致します。

#### a) 第 2 相が記録する追加情報

\* 第 2 相の、業務処理プログラム(COBOL プログラム)の中には、ファイルステータス(FILE STATUS)という名前の、2 バイトの(変数)領域が用意されています。この 2 バイトの領域は、1 バイトずつの領域が、2 つ組み合わせられた形になっています。

この 1 バイトの領域の 1 つは“状態キー1”と呼ばれ、もう一方の 1 バイトの領域は“状態キー2”と呼ばれます。

FILE STATUS	
状態キー1	状態キー2

\* 業務処理プログラムが、ISAM ファイルにアクセスすると、その結果を、COBOL のランタイムプログラムが、FILE STATUS の 2 バイトの変数領域の中に書き込みます。

\* ISAM ファイルのアクセスが成功すると、FILE STATUS の 2 バイトの変数領域のうちの、“状態キー1”に、0 が文字として書き込まれます。

(=状態キー1に実際に書き込まれる値は、16 進表示で 30)

\* 状態キー1に書き込まれた値が 0 以外の値になった場合は、何らかのエラーが発生しているので、業務処理プログラムは、状態キー1 と状態キー2 の値を、エラー情報として、ログファイルの中に書き込みます。

\* このとき、業務処理プログラムは、FILE STATUS の 2 バイトのそれぞれに、バイナリ値がセットされているものとして、そのバイナリ値を文字に置き換えて、追加情報として、“ Status = nnmm”という形式で、書き込みを行います。

このとき、nn には、状態キー1 の値、mm には、状態キー2 の値を書き込みます。

n および、m は、それぞれ 1 バイトを表すので、ログに記録されている Status 値は 4 バイトになります。

〔例〕既にオープンしている ISAM ファイルに、OPEN 文を実行しようとした場合。

- ① COBOL のランタイム プログラムは、状態キー1 に、(文字)4、状態キー2 に、(文字)1 をセットします。(下の表をご参照下さい。)
- ② この文字をバイナリ値と見なします。
- ③ 各 4 ビットを、数値として見ます。
- ④ その各 4 ビットに、対応する文字を割り当てます。(この段階で、4 ビットに 1 バイトが対応します。)
- ⑤ 各文字を、ログに書き込みます。  
nn には、0x33 0x34 = “34”  
mm には、0x33 0x31 = “31”  
が書き込まれます。
- ⑥ 文字として表示されると、“3431”になります。

ステップ	作 業	状態キー1		状態キー2	
①	COBOL ランタイム プログラムがセット	'4'		'1'	
②	バイナリ表現(16 進)	0x34		0x31	
③	各 4 ビット(nibble)の値	0x03	0x04	0x03	0x01

④	ソフトウェア部品システムで変換後 (各 nibble を、文字に変換)	0x33	0x34	0x33	0x31
⑤	ログに記入	0x330x340x330x31			
⑥	(文字として印刷されると)	"3431"			

\* このような複雑なことをしないで、COBOL のランタイム プログラムが、状態キーに書き込んだ値（上の例の、“41”）を、そのままログに書き込めばよいように思えますが、COBOL のランタイム プログラムは、

状態キー1 が、(文字)9 の場合は、状態キー2 には、バイナリ値をセットする  
ようになっています。(MicroFocus COBOL の仕様)

つまり、mm はバイナリ値になります。この値をログファイルに書き込んだとしても、ログファイルをテキストファイルとして開いたときに、mm に設定された値は（一般的には）読めません。そのため mm を、バイナリ値ではなく、文字に変換して、(2 という数値の場合は、文字の '2' に変換して) 文字列としてログファイルに書き込んでいます。

(バイナリ値を、4 ビットごとに切り分ければ、0x00~0x0F の範囲に入りますから、それぞれを文字に変換しておけば、テキストとして取り扱うことができます。)

\* 状態キー1 が、(文字の)9 となる例としては、ISAM ファイルのインデックスファイルが失われている場合があります。この場合は、状態キー2 には、0x2B がセットされますから、ログに記録される内容は、次のようになります。

[例]追加情報 : Program = MMES0020 D:¥BSS\_MEST¥dat¥MES\_BUN  
Status = 392B

状態キー2 の値が文字列として記録されています。このとき、状態キー1 の値(文字 '9'、バイナリ表示では、0x39)も、変換されるので、nn が文字列“39”として記録されます。

#### b) 第2相の Status の解読方法

\* エラーが発生して、ログファイルを見ると、業務処理プログラムの実行中にエラーが生じていることがわかり、“Status = ”に値が書き込まれていることが判ったとします。

このとき、実際には、どのようなエラーが発生したのかを知るには、次のようにします。

① “Status = ”に続く、4 つの文字を読み取ります。(ここでは仮に“PQRS”であったとします。)

② 4 つの文字を左右、2 つずつの組に分けます。(“PQ”と“RS”。“PQ”が状態キー1、“RS”が状態キー2、にそれぞれ対応しています。)

ここで、“PQ”が、“39”か、“39”以外かで、対応が分かります。

##### A) “PQ”が、“39”の場合

③ “RS”を16進の数と見なして、10進数に変換します。例えば、“RS”が、“2B”の場合は、2Bをそのまま16進数として受け取り、10進数に変換します。(2Bの場合は、10進数表示は、43になります。)

④ MicroFocus COBOL の、エラーメッセージを記載しているマニュアルを参照して、ランタイム・システム・エラーの項を開き、当該の番号のエラーの内容を調べます。

“RS”が、“2B”の場合は、番号43のメッセージを探します。(「索引ファイルにエラー情報がない」、という項目が見つかります。)

B) “PQ” が、“39” 以外の場合

③ この場合には、“PQ” は、“31”、“32”、“33”、“34”、の 4 種類しかありません。

④ 数値 0 は、文字としては “30” となりますから、文字から数値に戻すために、30 を引きます。残った値が、エラー発生時の状態キー1 の値となっています。

(“PQ” が、“31” の場合は、“30” を引きます。1 が残りますが、これが状態キー1 の値が 1 だったことを表します。)

⑤ “RS” についても、同様に、数値に戻します。これが状態キー2 の値です。

⑥ MicroFocus COBOL の言語リファレンスのマニュアルを開き、「ファイル入出力の概要」の項を開きます。

⑦ 状態キー1 の値毎に、エラーが分類されているので、状態キー2 の値の項を見つけます。例えば、Status = 3339 の場合は、“PQ” が “33”、“RS” が “39” になりますから、エラーの発生時は、状態キー1 の値が 3、状態キー2 の値が 9 だったことがわかります。

マニュアルを開いて、状態キー1 の値が “3” の場合を探すと、“永続誤り条件” の項が見つかります。その項のうちの、状態キー2 が、9 の場合を探します。(見つかった内容の記述が長いので、ここには記載しません。)

c) 第 3 相が記録する追加情報

\* 第 3 相のエラーは、主として、DB アクセス関数の内部で検出されます。

\* ソフトウェア部品システムでは、エラーの発生を検出する方法として、SQL 通信構造体 (sqlca: SQL Communications Area) を組み込む方法を採用しています。

DB アクセス関数の実行時に、SQL 文が Oracle<sup>[\*1]</sup> によって処理されると、処理の結果が、(この)sqlca 構造体の中にセットされます。

[\*1] SQL Server の場合も、同様に、処理の結果が、sqlca に記録されます。

\* sqlca 構造体には、9 つのメンバーが定義されていますが、ソフトウェア部品システムでは、sqlca 構造体のメンバーのうちの、sqlcode を参照します。

\* sqlca.sqlcode には、最後に実行された SQL 文のステータス・コード (SQL 処理の結果) が格納されています。コードの値が、正の値、0、負の値、のいずれであるか、によって意味は、それぞれ次のようになります。

[1] コードの値が、0 の場合。

エラーまたは例外の検出なしで、文が実行された。

[2] コードの値が、正の場合。

文は実行されたものの、例外が検出された。

[3] コードの値が、負の場合。

データベースまたはシステム、ネットワーク、アプリケーションのエラーが原因で、文が実行されなかった。

(データベースにとっては、このエラーは致命的なものであるため、ソフトウェア部品システムでは、このエラーの場合に、エラーログに記録をします。)

\* ソフトウェア部品システムでは、エラーログに記録をする場合、“The sqlcode is ” という固定の文言を使い、その後、sqlca.sqlcode の値を、“[]” で挟んで、ログに書き込みます。

[例]追加情報 : The sqlcode is [-1033].

d) 第 3 相の sqlcode の解読方法

- \* エラーが発生して、ログファイルを見ると、追加情報の項に、“The sqlcode is ” という文があり、DB アクセス関数でエラーが生じていることがわかったとします。
- \* 実際に、どのようなエラーが発生したのかを知るには、次のようにします。
  - [1] まず、括弧 “[ ] ” の中の値を読み取ります。
  - [2] 負の値になっているはずなので、符号を取ります。
  - [3] 動作をしている DB に付属しているマニュアルのうちの、エラー・メッセージを解説しているマニュアルを開いて、(符号をとった番号(コード)に対応する) エラーの項目を探します。
- \* 例えば、DB が正常に動作している状態で、DB のテーブルの 1 つをドロップします。この状態で、その DB のテーブルにアクセスすると、DB が Oracle の場合は、エラーログとして、次のような追加情報が、エラーログに書き込まれます。

[例]追加情報 : The sqlcode is [-942].

- [1] sqlcode は、負の数なので、SQL 文が実行できず、エラーになったことがわかります。
- [2] 符号を除くと、942 になります。
- [3] Oracle のメッセージの解説をしているマニュアルを開いて、この番号を探すと、“ORA-00942” という項があります。その内容は、“表、またはビューが存在しません。” となっていて、“テーブルがドロップされている” という原因に対応しています。

[注] なお、sqlcode の数値に、どのようなエラーに対応しているかは、DB のメーカーによって異なるので、エラーの具体的な内容を知るには、使用している DB のメーカーから発行されたマニュアルを参照する必要があります。

#### ご注意

- ◎ 本書は著作権法で保護されている著作物です。
- ◎ 本書の内容の一部または全部をソフトウェア部品の普及の目的に限り複製することを認めます。
- ◎ 本書の内容は予告なく変更される場合があります。
- ◎ 本書は内容について万全を期して制作いたしましたが、万一、ご不審な点や誤り、記載漏れなどお気づきのことがありましたら、ご連絡下さい。

---

## ソフトウェア部品システムのエラーログ

---

発行者 清川茂満  
発行所 株式会社セントラルソフトサービス Q&A センター  
〒 444-0831 岡崎市羽根北町 2 丁目 1 番 8  
TEL: 0564-59-3221 e-mail : QAcenter@css-snet.co.jp  
発行日 2008 年 11 月 30 日

---

Copyright © 2008 CSS Co.,Ltd. Allrights reserved.

---