中信银行 CHINA CITIC BANK

中信银行核心系统升级项目

核心系统 RPG 语言编码规范

发布日期	
控制级别	内部资料
制定部门	

❒ **文档属性**

属性	内容
◆ 用户名称：	◆
◆ 文档标题：	◆
◆ 文档编号：	◆
◆ 版本日期：	◆
◆ 发布版本：	◆
◆ 适用范围：	◆
◆ 作者：	◆

❒ **文档审批**

审批人	职务	审批时间	审批意见	发文编号
◆	◆	◆	◆	◆
◆	◆	◆	◆	◆

❒ **修订内容**

版本	修正章节	日期	修正人	变更记录
◆	◆	◆	◆	◆
◆	◆	◆	◆	◆
◆	◆	◆	◆	◆
◆	◆	◆	◆	◆

❒ **模板修订历史**

版本	生效时间	变更概要	作者	审核	批准
◆	◆	◆	◆	◆	◆
◆	◆	◆	◆	◆	◆
◆	◆	◆	◆	◆	◆
◆	◆	◆	◆	◆	◆

目录

1 基本原则

1.1 基本要求

1) 程序结构清淅，简单易懂，主程序及程序处理分支裁剪合理。
2) 目的简单明确，直接了当，代码精简，避免垃圾程序。
3) 尽量使用系统 API、公共模块、公共函数或公共子程序完成公共操作。
4) 采用 ILE RPG 编码，编码格式使用自由格式。ILE RPG 在程序建立、程序管理、程序调用、源码的可读性、调用 API 等方面比较 RPG 固定格式编码有了很大的提高及改进。
5) 源码必须有 TEXT，TEXT 对源码进行准确的功能概述。生成的目标也必须有 TEXT，TEXT 对目标进行送爽的功能概述。如果目标是由源码生成，则目标的 TEXT 应该与源码的 TEXT 一致。

1.2 可读性要求

1) 每个源程序开始都有说明源成员说明，说明内容必须项目规范编写。内容包括：公司名称、系统名称、程序名、功能描述、创建/维护程序员姓名、创建/维护时间、创建/维护说明、输入/输出参数说明、引用文件及使用方法说明、子过程说明。复杂的算法需要加上流程说明。
2) 程序变量（结构、数组、数据区域等）定义或引用时，注释能反映其含义。
3) 常量定义（CONSTANT）有相应说明。
4) 处理过程的每个代码段需要有相关注释说明。
5) 在典型算法前都有注释说明。
6) 保持注释与代码的一致。
7) 利用空行、注释行间隔体现程序逻辑结构。
8) 每行有效代码长度一般不超过 70 个字符长，以保证一屏内可以显示每行

完整代码内容，遇长语句（>70 字符）时应分成多行书写。

9) 循环、分支层次一般不要超过五层。自由格式中，每一层次代码的开始位置退 3 个空格，使程序层次更加清淅，避免层次错误。

10) 注释语句位于所注释代码的上方。

11) 简单的语句不需要注释。

12) 注释行数（不包括源码说明部分）约占总行数的 1/5～1/3 为宜。

13) 源码维护时增加的注释，应该包括：维护项目代码、维护时间、开发人员等内容。

1.3 结构化要求

1) 遵守结构化程序设计的规定。

2) 自由格式 RPG 已经不再支持 GOTO 语句、不支持使用指示器（第 10－12 位）控制命令的执行。

3) 所有的逻辑判断使用表达式。

4) 使用 IF/ELSE/ENDIF 语句进行条件处理。

5) 使用 SELECT/WHEN/OTHER/ENDSL 语句实现多路分支。避免不必要的分支。

6) 使用 DOW/ENDDO、DOU/ENDDO、FOR/ENDFOR 语句实现循环处理。避免从循环引出多个出口。

1.4 正确性与容错性要求

1) 程序需要在开始阶段对变量进行初始化，保证后续流程可以正常使用。

2) 程序需要对文件的指针进行初始化，保证后续流程可以正常操作。

3) 程序需要对输入数据进行必要的合法性检查。

4) 程序处理过程中需要考虑环境、状态的关系。在程序增加必要的判断，并对可能的异常增加主动处理。例如：文件锁冲突、键字重复等。

5) 程序编写后，应该检查的逻辑流程。保证顺序、条件、循环的处理逻辑、

层次正确。

6）修改程序前，需要先检查对其他程序的影响。

7）单元测试也是编程的一部份，提交联调测试的程序必须通过单元测试。

1.5 可重用性要求

1）重复使用的完成相对独立功能的算法或代码应抽象为公共模块、公共函数、公共子程序。

2）公共模块、公共函数、公共子程序应考虑面向对象思想，减少外界联系。

3）公共模块、公共函数应该按功能组装成不同的服务程序。程序编译时，不能直接联编公共模块、公共函数，只能联编服务程序。

4）需要提供公共模块、公共函数、公共子程序的使用模板。

2 排版

2.1 RPG 程序的结构

RPG 程序由以下几部分组成：

1) 源成员说明：源成员说明，说明内容必须项目规范编写。内容包括：公司名称、系统名称、程序名、功能描述、创建/维护程序员姓名、创建/维护时间、创建/维护说明、输入/输出参数说明、引用文件及使用方法说明、子过程说明。复杂的算法需要加上流程说明。
2) 控制信息定义（H 表），这部分内容是可选的，不必须定义。
3) 文件定义（F 表），如果程序不使用文件可以不定义。定义内容：
   a) 定义程序使用的文件，包括：物理文件、逻辑文件、显示文件、打印文件。
   b) 定义文件的使用方法，包括：打开方式、事务处理控制、异常控制、实际使用的文件及成员。
4) 数据定义（D 表），定义内容：
   a) 程序原型定义，包括程序参数、参数类型等。由于自由格式不支持 *ENTRY 方式接收参数，必须使用程序原型定义接收参数。
   b) 调用程序原型定义，定义调用的程序名称、参数、参数类型。其中程序名称可以是变量，在程序执行过程中获得程序名称。由于自由格式不支持 CALL、CALLB 调用程序或模块，必须定义程序原型，在程序中使用 CALLP（可以省略）调用程序或模块。
   c) 定义数据结构，可以定义参考外部文件定义数据结构、键字结构。由于自由格式不支持 PLIST 方式定义键字列表，只能使用定义键字结构的方式。
   d) 定义程序变量，除控制程序流程的变量（例如：循环控制变量、条件变量）外，都应该参考数据字典或数据文件字段定义。
   e) 定义使用的常量。
5) 输入定义（I 表），这部分内容是可选的，不必须定义。主要用于定义程

序描述的文件结构。一般情况下，应用程序都应该使用外部描述方式定义文件结构。

6) 程序处理（C 表），自由格式不再使用第 6 位为 C 的编程方式。定义内容包括：

a) 代码开始定义：/FREE

b) 程序的主过程定义：主过程应该只描述程序的主处理逻辑，使主过程简单、明了。

c) 程序的子例程定义：将程序中不同功能划分多个的子例程，每个子例程完成一定的功能。主过程可以调用子例程，子例程也可以调用子例程。子例程以 BEGSR 开始，以 ENDSR 结束。子例程不能嵌套定义。

d) 在自由格式中可以使用嵌入 SQL，语法为 EXEC SQL。

e) 代码结束定义：/END-FREE

7) 输出定义（O 表），一般输出应该使用 PRTF 定义，应该不需要使用输出定义。

8) 过程定义，程序使用的过程定义。一个程序中可以定义多个过程。如果是公用过程，应用抽取出来，建立公用过程，再组装成服务程序，程序与服务程序联编。程序使用过程的方法基本相同，但可以大量减少代码量、维护工作量。定义方法：

a) 过程开始定义：P……B

b) 过程变量定义，在过程中使用 D 表定义过程原型定义、调用其他程序/过程原型定义、数据结构、变量、常量。

c) 过程处理定义，使用自由格式描述过程处理流程。以/FREE 开始，以/END-FREE 结束。

d) 过程结束定义：P……E

2.2 程序处理部分的书写要求

1) 程序代码段要采用空行分隔方式书写。相对独立的代码段之后必须加空行。

2) 一行程序以小于 70 字符为宜，不要写得过长，以保证在一屏中可以看到

完整的程序语句，避免错误理解程序逻辑。

3) 较长的语句（>70 字符）要分成多行书写，长表达式要在低优先级操作符处划分新行，操作符放在上一行结尾，避免错误分行。

4) 划分出的新行需要缩进 3 个空格，使排版整齐，增强可读性。

5) 在两个以上的关键字、变量、常量进行对等及非对等操作时，它们之间的操作符之前、之后或者前后要加一个空格。

6) 语句结束符‘;’必须与语句的最后一个非空字符间保留一个空格。

7) 主过程的语句比/FREE 的‘/’缩进 3 个空格（即从第 10 个位置开始编写），同一层次语句首字母对齐。

8) 如果一个语句不能在一行完整写完，可以换行继续编写，不需要换行符，一般将操作符保留在行尾。换行后首字母比第一行首字母缩进 3 个空格。

9) 一个语句可以写多个续行，所有续行的首字母对齐。

10) 循环、分支层次一般不要超过五层。每一层次代码较上一层次缩进 3 个空格，使程序层次更加清淅，避免层次错误。

11) 不同处理逻辑的代码段间保留一行空行。

12) 子例程的开始语句（BEGSR）首字母比/FREE 的‘/’缩进 3 个空格（即从第 10 个位置开始写）；子例程的结束语句（ENDSR）首字母应该与开始语句对齐。

13) 子例程内的第一行语句与子例程序的开始语句（BEGSR）保留一行空行。

14) 子例程结束语句（ENDSR）与的子例程的最后一行语句保留一行空行。

15) 子例程内的语句，应该比开始语句（BEGSR）缩进 3 个空格。

16) 循环、分支层次一般不要超过五层。每一层次代码较上一层次缩进 3 个空格，使程序层次更加清淅，避免层次错误。

程序样例：

```
S_RECNUM10 = 1 ;
DOW S_RECNUM10 <= S_RECNUM ;
   CHAIN S_RECNUM10 OPTSFL10 ;
   IF CPAVALC <> *BLANK AND %SUBST(CPBVALC:1:1) = '&' AND
```

```
      CPCPSTR <> *ZERO ;
      %SUBST(CPAPARM:CPCPSTR:20) = CPAVALC ;
   ENDIF ;
   S_RECNUM10 = S_RECNUM10 + 1 ;
ENDDO ;
```

3 注释

3.1 源码说明的方法

3.1.1 RPG程序源码说明方法

3.1.1.1 版权信息

第 6 位为空，第 7 位为*，第 70 位为*，中间为信息内容。定义方式如下：

```
     ****************************************************************
     ****************************************************************
     ** (C) Copyright  -   中信银行   CHINA CITIC BANK              *
     **                  Year 2005, all rights reserved.          **
     ****************************************************************
     ****************************************************************
```

3.1.1.2 源码基本信息

与版权信息保留一行空行。

第 6 位为空，第 7 位为*，第 70 位为*，中间为信息内容，包括：系统名称（统一为中信银行核心系统））、程序名称、功能描述、程序员名、编程日期。定义方式如下：

```
     **BBB************************************************************
     *                                                              *
     *    系统名称   : 中信银行核心系统                             *
     *    程序名称   : PROGRAMxxx                                   *
     *    功能描述   : （程序描述，应该与源成员的 TEXT 相同）       *
     *    程序员名   : （开发人员中文姓名）                         *
     *    编程日期   : YYYY/MM/DD                                   *
     *                                                              *
     ****************************************************************
```

3.1.1.3 使用文件说明

说明程序使用的文件，处理方式分为 I（读）、O（写）、U（更新）、C（显示文件专用），I/O/U 可以组合使用。如果一个程序使用多个文件，需要逐行说明。

```
*****************************************************************
*                                                               *
*    文件名       文件描述                          处理方式     *
*   ----------  ------------------------------  ----------      *
*   FILENAMExx   （文件描述，与 OBJ 的 TXT 相同）   I           *
*                                                               *
*****************************************************************
```

3.1.1.4 程序参数说明

说明程序的接口参数，处理方式分为 I（输入）、O（输出）、I/O（输入输出）。如果一个程序有多个参数，需要逐行说明。

```
*****************************************************************
*                                                               *
*    参数         参数描述                          处理方式     *
*   ----------  ------------------------------  ----------      *
*   TWA          交易数据区                        I           *
*   XXXXXX       组件数据区                        I/O         *
*                                                               *
*****************************************************************
```

3.1.1.5 子例程说明

说明程序中使用的子例程。如果一个程序有多个子例程，需要逐行说明。

```
*****************************************************************
*                                                               *
*    子例序       子例序描述                                    *
*   ----------  ----------------------------------------        *
*   @SR_xxxxxx   （子例程序说明）                               *
*                                                               *
**EEE************************************************************
```

3.1.1.6 程序维护说明

如果修改程序内容，需要在程序说明中增加修改说明，每次修改都需要增加。定义方法如下。

```
*****************************************************************
*                                                               *
*    项目编号   :  维护项目编号
*    维护说明   :  （修改功能描述）                             *
```

```
*     维护人员   : （开发人员中文姓名）                              *
*     维护日期   : YYYY/MM/DD                                       *
*                                                                   *
**EEE***************************************************************
```

3.1.2程序源码说明的标准样例

不类型的源码，说明方式略有差异。但基本内容与 RPG 的说明方式相同。以下分别给出不同类型原码的说明样例。

3.1.2.1 RPGLE、SQLRPGLE程序源码说明样例

```
*********************************************************************
*********************************************************************
** (C) Copyright  -   中信银行   CHINA CITIC BANK                   *
**                   Year 2005, all rights reserved.               **
*********************************************************************
*********************************************************************

**BBB****************************************************************
*                                                                   *
*     系统名称   : 中信银行核心系统                                 *
*     程序名称   : PROGRAMxxx                                       *
*     功能描述   : （程序描述，应该与源成员的 TEXT 相同）           *
*     程序员名   : （开发人员中文姓名）                             *
*     编程日期   : YYYY/MM/DD                                       *
*                                                                   *
*********************************************************************
*                                                                   *
*     文件名       文件描述                          处理方式       *
*   ----------  ------------------------------  ----------         *
*   FILENAMExx   （文件描述，与 OBJ 的 TXT 相同）   I               *
*                                                                   *
*********************************************************************
*                                                                   *
*     参数         参数描述                          处理方式       *
*   ----------  ------------------------------  ----------         *
*   TWA          交易数据区                      I                 *
*   XXXXXX       组件数据区                      I/O               *
*                                                                   *
*********************************************************************
*                                                                   *
*     子例序       子例序描述                                       *
```

```
*   ----------  ----------------------------------------      *
*   @SR_xxxxxx   （子例程序说明）                              *
*                                                             *
**EEE***********************************************************
```

3.1.2.2 CLLE、C程序源码说明样例

```
/*************************************************************/
/*************************************************************/
/** (C) Copyright  -   中信银行   CHINA CITIC BANK           */
/**                    Year 2005, all rights reserved.       **/
/*************************************************************/
/*************************************************************/

/**BBB********************************************************/
/*                                                           */
/*    系统名称   : 中信银行核心系统                          */
/*    程序名称   : PROGRAMxxx                                */
/*    功能描述   : （程序描述，应该与源成员的 TEXT 相同）    */
/*    程序员名   : （开发人员中文姓名）                      */
/*    编程日期   : YYYY/MM/DD                                */
/*                                                           */
/*************************************************************/
/*                                                           */
/*    文件名       文件描述                    处理方式      */
/*   ----------  ----------------------------  ----------    */
/*   FILENAMExx   （文件描述，与 OBJ 的 TXT 相同）  I        */
/*                                                           */
/*************************************************************/
/*                                                           */
/*    参数         参数描述                    处理方式      */
/*   ----------  ----------------------------  ----------    */
/*   TWA          交易数据区                  I             */
/*   XXXXXX       组件数据区                  I/O           */
/*                                                           */
/*************************************************************/
/*                                                           */
/*    子例序       子例序描述                                */
/*   ----------  ----------------------------------------    */
/*   @SR_xxxxxx   （子例程序说明）                           */
/*                                                           */
/**EEE********************************************************/
```

3.1.2.3 PF、LF、DSPF、PRTF源码说明样例

```
     ******************************************************************
     ******************************************************************
     ** (C) Copyright  -   中信银行   CHINA CITIC BANK                *
     **                   Year 2005, all rights reserved.           **
     ******************************************************************
     ******************************************************************

     **BBB***********************************************************
     *                                                                *
     *    系统名称   : 中信银行核心系统                               *
     *    文件名称   : PROGRAMxxx                                     *
     *    文件描述   : （文件描述，应该与源成员的 TEXT 相同）          *
     *    程序员名   : （开发人员中文姓名）                           *
     *    编程日期   : YYYY/MM/DD                                     *
     *                                                                *
     **EEE***********************************************************
```

3.1.2.4 BND源码说明样例

```
     /******************************************************************/
     /******************************************************************/
     /** (C) Copyright  -   中信银行   CHINA CITIC BANK                */
     /**                   Year 2005, all rights reserved.           **/
     /******************************************************************/
     /******************************************************************/

     /**BBB***********************************************************/
     /*                                                                */
     /*    系统名称   : 中信银行核心系统                               */
     /*    程序名称   : PROGRAMxxx                                     */
     /*    功能描述   : （程序描述，应该与源成员的 TEXT 相同）          */
     /*    程序员名   : （开发人员中文姓名）                           */
     /*    编程日期   : YYYY/MM/DD                                     */
     /*                                                                */
     /**EEE***********************************************************/
```

3.2 RPG 程序内的注释方法说明

3.2.1 注释基本原则及方法

1) 边写代码边注释，修改代码同时修改相应的注释，以保证注释与代码的一致性。注意删除不再有用的注释。
2) 注释的内容要清楚、明了，含义准确，防止注释二义性。
3) 避免在注释中使用缩写，特别是非常用缩写。
4) 注释内容应该放在对应代码的上方。
5) 对于所有有物理含义的变量、常量，所有数据结构声明（包括：数组、结构、数据区域等），如果其命名不是充分自注释的，在声明时都必须加以注释说明其物理含义，注释应放在其上方相邻位置。
6) 注释内容应该与注释对象列对齐。
7) 对变量的定义和分支语句（条件分支、循环语句等）必须编写注释。
8) 避免在整句代码或表达式的中间插入注释。
9) 通过对函数或过程、变量、结构等正确的命名以及合理地组织代码的结构，使代码成为自注释的。
10) 注释格式尽量统一。
11) 注释应考虑程序易读，尽量使用中文注释。

3.2.2 H表的注释方法

每行定义一个控制信息，如果控制信息不能直接反应控制功能，则需要在这一行之上增加注释，省略第 6 位的 H，第 7 位为*。

定义方法如下：

```
     * 允许调试，可以使用 DUMP
     H DEBUG(*YES)
```

3.2.3 F表的注释方法

在 F 表开始处，增加文件定义的说明，注释方法是统一的，注释必须与 H 表或源码说明保持一行空行。

注释省略第 6 位的 F，第 7 位为*，注释方法如下：

```
     *****************************************************************
     * 文件定义
     *****************************************************************
    FTLCPAL00  IF   E           K DISK
```

由于已经在源码注释中逐一说明了使用的文件，可以不再对使用文件进行注释。

3.2.4D表的注释方法

D 表可以定义多种类型不同的数据。每种类型的数据应该集中定义。每种类型之前增加定义说明注释。

注释应该与上一段代码间保持一行空行，省略第 6 位的 D，第 7 位为*。

程序原型定义的注释：

```
     *****************************************************************
     * 程序原型定义
     *****************************************************************
    D PGM_XXXXX       PR                  EXTPGM('SCXXXXXR')
    D  PR_TWA                             LIKE(XDS_SCCTWA)
    D  PR_XXXXX                           LIKE(DS_XXXXX)
    D PGM_DBGD2       PI
    D  C_TWA                              LIKE(XDS_SCCTWA)
    D  C_XXXXX                            LIKE(DS_XXXXX)
```

程序调用其他程序的原型定义注释：

```
     *****************************************************************
     * 调用程序原型定义
     *****************************************************************
    D PGM_XXXX1       PR                  EXTPGM('SCXXXX1R')
    D  PR_TWA                             LIKE(XDS_SCCTWA)
    D  PR_XXXX1                           LIKE(DS_XXXX1)

    D PGM_XXXX2       PR                  EXTPGM('SCXXXX2R')
    D  PR_TWA                             LIKE(XDS_SCCTWA)
    D  PR_XXXX2                           LIKE(DS_XXXX2)
```

数据结构及键字结构定义注释：

```
     ****************************************************************
     * 数据结构及键字结构定义
     ****************************************************************
D DS_zzyyy       E DS                    EXTNAME(zzyyyA)
D DS_zzyyy_K00   E DS                    EXTNAME(zzyyyL00:*KEY)
D                                        PREFIX(K00_)
```

变量定义注释：

```
     ****************************************************************
     * 变量定义
     ****************************************************************
D C_MSCD          S                      LIKE(@@MSCD)
```

常量定义注释：

```
     ****************************************************************
     * 常量定义
     ****************************************************************
D L_MSCD_OK       C                      'AAAAAAA'
```

3.2.5C表的注释方法

自由格式的注释方法，不再使用第 7 位*的方式。而是使用//，开始位置为第 8 位开始的任何位置。

注释应该对被注释语句的上方，开始位置与被注释语句对齐。不能一行写完整的注释，应该换行写。换行后仍然使用//开始，开始位置与被注释语句对齐。

子例程的注释：在子例程前（BEGSR 语句之上）进行子例程的注释，说明子例程的功能。注释行以//开始，与 BEGSR 的首字母对齐。

```
     /FREE
```

```
        // 执行数据初始化子例程
        EXSR @SR_DTAINI ;

        *INLR = *ON ;
        RETURN ;

       // ------------------------------------------------------
       //  错误处理子过程
       // ------------------------------------------------------
       BEGSR @SR_DTAINZ ;

          // 获 TWA 的指针
          PTR_TWA = %ADDR(C_TWA) ;

       ENDSR ;
    /END-FREE
```

3.2.6过程的注释方法

在每个过程前对过程进行说明注释，内容包括：过程名称、功能描述、输入参数描述、输出描述。

定义方法如下：

```
    ****************************************************************
    *  程序名称：  F_SCJNDAT
    *  功能描述：日志文件日期计算程序
    *  接口参数：
    *      01 *              日志文件日期计算数据结构
    *  返回参数：
    *     无
    ****************************************************************
```

过程内部的注释与主过程相同。

4 命名规则

4.1 程序命名规则

4.1.1 服务组件命名规则

交易服务组件命名规则：zznnnnt，其中：

zz：应用模块标识；

nnnn：交易码；

t：程序源码类型

其他服务组件命名规则：zzxxxxxxt，其中：

zz：应用模块标识；

xxxxxx：组件名称；

t：程序源码类型

程序源码类型：（与现有应用的规范保持一致）

R：RPGLE、SQLRPGLE 源码

L：CLLE 源码

C：C 程序源码

B：COBOL 程序源码

RPGLE、CLLE、C 程序（*PGM）源码存放在 CzzPGMSRC 源文件中；

SQLRPGLE 程序（*PGM）（如果进行事务控制）源码存放在 CzzPGMSRC 源文件中；

SQLRPGLE 程序（*PGM）（如果不进行事务控制）源码存放在 CzzSQLSRC 源文件中；

4.1.2单元组件命名规则

单元组件命名规则：zzxxxxxxt，其中：

zz：应用模块标识；

xxxxxx：组件名称；

t：程序源码类型

程序源码类型：（与现有应用的规范保持一致）

R：RPGLE、SQLRPGLE 源码

L：CLLE 源码

C：C 程序源码

B：COBOL 程序源码

RPGLE、CLLE、C 程序（*PGM）源码存放在 CzzPGMSRC 源文件中；

SQLRPGLE 程序（*PGM）（如果进行事务控制）源码存放在 CzzPGMSRC 源文件中；

SQLRPGLE 程序（*PGM）（如果不进行事务控制）源码存放在 CzzSQLSRC 源文件中；

4.1.3函数组件命名规则

函数组件命名规则：zzxxxxxxt，其中：

zz：应用模块标识；

xxxxxx：组件名称；

t：程序源码类型

程序源码类型：（与现有应用的规范保持一致）

R：RPGLE、SQLRPGLE 源码

L：CLLE 源码

C：C 程序源码

B：COBOL 程序源码

RPGLE、CLLE、C 程序（*PGM）源码存放在 CzzPGMSRC 源文件中；

SQLRPGLE 程序（*PGM）（如果进行事务控制）源码存放在 CzzPGMSRC 源文件中；

SQLRPGLE 程序（*PGM）（如果不进行事务控制）源码存放在 CzzSQLSRC 源文件中；

RPGLE、CLLE、C 模块（*MODULE）源码存放在 CzzMODSRC 源文件中；

SQLRPGLE 模块（*MODULE）（如果进行事务控制）源码存放在 CzzMODSRC 源文件中；

SQLRPGLE 模块（*MODULE）（如果不进行事务控制）源码存放在 CzzSQLMOD 源文件中；

4.2 组件接口文件命名规则

组件接口的定义方式为：

先建立组件接口结构的物理文件，在程序中使用参考文件结构的方式定义组件接口数据结构。

组件接口文件源码应该保存在源文件 CzzIOFSRC 中。

组件接口文件目标应该保存在 CNCBIOFD 中。

4.2.1 交易服务组件接口文件

交易服务组件接口分为:输入接口、输出接口、多页查询输出格式定义。

4.2.1.1 输入接口文件

输入接口文件名称（不循环字段）：zznnnnIS，前 6 位应该与组件程序名称的前 6 位相同，格式名称为：RnnnnIS，字段名称 ISxxxx；

输入接口文件名称（循环字段）：zznnnnIM，前 6 位应该与组件程序名称的前 6 位相同，格式名称为：RnnnnIM，字段名称 IMxxxx；

文件名称中：

zz：应用模块标识；

nnnn：交易码；

字段名称中：

xxxx 为字段名称，与数据字典名称相同；

IS、IM 为字段前缀，表示输入接口字段；

4.2.1.2 输出接口文件

输出接口文件名称（不循环字段）：zznnnnES，前 6 位应该与组件程序名称的前 6 位相同，格式名称为：RnnnnES，字段名称 ESxxxx；

输出接口文件名称（循环字段）：zznnnnEM，前 6 位应该与组件程序名称的前 6 位相同，格式名称为：RnnnnEM，字段名称 EMxxxx；

文件名称中：

zz：应用模块标识；

nnnn：交易码；

字段名称中：

xxxx 为字段名称，与数据字典名称相同；

ES、EM 为字段前缀，表示输出接口字段；

4.2.1.3 多页查询输出结构文件

输出接口文件名称：zznnnnEL，前 6 位应该与组件程序名称的前 6 位相同，格式名称为：RnnnnEL，字段名称 ELxxxx；

文件名称中：

zz：应用模块标识；

nnnn：交易码；

字段名称中：

xxxx 为字段名称，与数据字典名称相同；

EL 为字段前缀，表示多页查询输出结构字段；

4.2.2其他服务组件接口文件

其他服务组件接口分为:输入接口、输出接口

4.2.2.1 输入接口文件

输入接口文件名称（不循环字段）：zzxxxxxxIS，前 8 位应该与组件程序名称的前 8 位相同，格式名称为：RxxxxxxIS，字段名称 ISxxxx；

输入接口文件名称（循环字段）：zzxxxxxxIM，前 8 位应该与组件程序名称的前 8 位相同，格式名称为：RxxxxxxIM，字段名称 IMxxxx；

文件名称中：

zz：应用模块标识；

xxxxxx：组件名称；

字段名称中：

xxxx 为字段名称，与数据字典名称相同；

IS、IM 为字段前缀，表示输入接口字段；

4.2.2.2 输出接口文件

输出接口文件名称（不循环字段）：zzxxxxxxES，前 8 位应该与组件程序名称的前 8 位相同，格式名称为：RxxxxxxES，字段名称 ESxxxx；

输出接口文件名称（循环字段）：zzxxxxxxEM，前 8 位应该与组件程序名称的前 8 位相同，格式名称为：RxxxxxxEM，字段名称 EMxxxx；

文件名称中：

zz：应用模块标识；

xxxxxx：交易码；

字段名称中：

xxxx 为字段名称，与数据字典名称相同；

ES、EM 为字段前缀，表示输出接口字段；

4.2.3单元组件、函数组件接口文件

单元组件、函数服务组件接口分为:输入接口、输出接口

4.2.3.1 输入接口文件

输入接口文件名称（不循环字段）：zzxxxxxxIS，前 8 位应该与组件程序名称的前 8 位相同，格式名称为：RxxxxxxIS，字段名称 ISxxxx；

输入接口文件名称（循环字段）：zzxxxxxxIM，前 8 位应该与组件程序名称的前 8 位相同，格式名称为：RxxxxxxIM，字段名称 IMxxxx；

文件名称中：

zz：应用模块标识；

xxxxxx：组件名称；

字段名称中：

xxxx 为字段名称，与数据字典名称相同；

IS、IM 为字段前缀，表示输入接口字段；

4.2.3.2 输出接口文件

输出接口文件名称（不循环字段）：zzxxxxxxES，前 8 位应该与组件程序名称的前 8 位相同，格式名称为：RxxxxxxES，字段名称 ESxxxx；

输出接口文件名称（循环字段）：zzxxxxxxEM，前 8 位应该与组件程序名称的前 8 位相同，格式名称为：RxxxxxxEM，字段名称 EMxxxx；

如果全部字段都是循环字段，则只定义 zznnnnEM；

文件名称中：

zz：应用模块标识；

xxxxxx：交易码；

字段名称中：

xxxx 为字段名称，与数据字典名称相同；

ES、EM 为字段前缀，表示输出接口字段；

为了能够正确、完整地返回信息，在输出接口应该定义返回码 ESMGID、附加信息 ESAINF。

4.3 数据结构命名规则

4.3.1 参考外部文件的结构命名规则

DS_zzxxx

DS_：表示数据结构

zzxxx：与文件名称的前 5 位相同。

4.3.2 文件键字结构命名规则

参考物理文件定义的键字结构名称为：DS_zzxxx_K

DS_：表示数据结构

zzxxx：与文件名称的前 5 位相同

_K：表示键字

键字结构的字段使用 PREFIX 增加前缀 K_。

参考逻辑文件定义的键字结构名称为：DS_zzxxx_Knn

DS_：表示数据结构

zzxxx：与文件名称的前 5 位相同

_Knn：表示键字，nn 与逻辑文件的序号相同

键字结构的字段使用 PREFIX 增加前缀 Knn_。

4.3.3 公用数据结构的命名规则

XDS_xxxxxx

XDS_：表示公用数据结构

xxxxxx：数据结构功能描述

4.3.4程序数据结构的命名规则

DS_xxxxxx

DS_：表示数据结构

xxxxxx：数据结构功能描述

4.3.5组件接口的命名规则

4.3.5.1 交易服务组件接口数据结构命名规则

输入接口数据结构：zznnnnI

输出接口数据结构：zznnnnE

多页查询输出数据结构：zznnnnEL

其中：

zz：应用模块标识；

nnnn：交易码；

4.3.5.2 其他服务组件接口数据结构命名规则

组件接口数据结构：zzxxxxxx

其中：

zz：应用模块标识；

xxxxxx：组件名称；

4.3.5.3 单元组件、函数组件接口数据结构命名规则

组件接口数据结构：zzxxxxxx

其中：

zz：应用模块标识；

xxxxxx：组件名称；

4.3.6技术平台数据区及接口的命名规则

技术平台数据区结构定义与组件接口定义方法相同，也是先定义文件结构，再参考文件定义数据区结构，但指针类型不能在文件中定义，直接在数据结构中

定义。

数据区	数据区说明	结构文件	接口名称	标准数据结构
TWA	交易数据区	SCTWAA	SCCTWA	XDS_SCCTWA
TIA	交易输入区	SCTIAA	SCCTIA	XDS_SCCTIA
TOA	交易输出区	SCTOAA	SCCTOA	XDS_SCCTOA
SCA	技术参数区	SCSCAA	SCCSCA	XDS_SCCSCA
……				

4.4 变量命名规则

4.4.1 公用变量的命名规则

变量名称：Xz_yyyy

X 表示公用变量，

z 为字段类型

yyyy 为字段名称,与对应的数据字典相同

字段类型分为：

B 二进制型，对应 B 型，一般 2 个字节的二进制数定义为 5I0,4 个字节的二进制数定义为 10I0；

C 字符型，对应 A/0 型；

P 压缩数字，对应 P 型；

S 区位数字，对应 S 型；

D 日期型，对应 D 型；

T 时间型，对应 T 型；

Z 时间戳，对应 Z 型；

I 指示器，对应 N 型，逻辑型；

4.4.2 程序变量的命名规则

变量名称：z_yyyy

z 为字段类型

yyyy 为字段名称。

字段类型分为：

B 二进制型，对应 B 型，一般 2 个字节的二进制数定义为 5I0,4 个字节的二进制数定义为 10I0；

C 字符型，对应 A/0 型；

P 压缩数字，对应 P 型；

S 区位数字，对应 S 型；

D 日期型，对应 D 型；

T 时间型，对应 T 型；

Z 时间戳，对应 Z 型；

I 指示器，对应 N 型，逻辑型；

4.5 常量命名规则

4.5.1 公用常量的命名规则

常量名称：G_xxxx_yyyy

G_表示公用常量

xxxx 为常量类别，一般与常量对应的数据字典名称相同

yyyy 为常量代码，对应常量的取值

4.5.2 程序常量的命名规则

常量名称：L_xxxx_yyyy

L_表示程序定义的本程序内使用的常量

xxxx 为常量类别，一般与常量对应的数据字典名称相同

yyyy 为常量代码，对应常量的取值

4.6 子例程命名规则

子例程的命名规则：@SR_xxxxxx

@SR_：表示子例程

xxxxxx：表示子例程名称。

4.7 服务程序命名规则

4.7.1 服务程序的名称

服务程序命名规则：zzPUBSRVn

zz：应用模块标识；

PUBSRV：表示服务程序，固定使用；

n：服务程序序号；

服务程序源码保存在源文件 CzzSRVSRC 中，zz 为应用模块标识。

4.7.2 组成服务程序的*MODULE名称

按一般程序的命名规则命名。

4.7.3 组成服务的功能函数名称

服务程序的函数命名规则：F_zzxxxxxx

F_：表示函数；

zz：应用模块标识；

xxxxxx：*MODULE 名称，不包含应用模块标识、源码类型

4.7.4 组成服务程序联编定义的名称

服务程序联编定义的名称：zzPUBSRVnB

zz：应用模块标识；

PUBSRV：表示服务程序；

n：服务程序号

B：表示联编定义，源码类型为 BND。

服务程序源码保存在源文件 C@@BNDSRC 中，应用系统共用一个源文件。

4.7.5服务程序函数原型定义的名称

服务程序函数原型：zzPUBSRVnH，

zz：应用模块标识；

PUBSRV：表示服务程序，

n：服务程序号，

H：表示原型定义，源码类型为 RPGLE。

服务程序源码保存在源文件 C@@DFNSRC 中，应用系统共用一个源文件。

5 F 表定义方法

5.1 数据文件定义说明

5.1.1 文件基本定义方法说明

一般的文件操作的方式分为：

I：输入方式，只读方式，不允许更新；

U：更新方式，包括含输入方式，读数据时会对数据加锁；

O：输出方式，只写方式，不允许读数据；

增加数据：定义增加方式为 A，O 方式不需要定义。

文件格式：一般定义为 E，表示使用外部文件描述。

随机访问：定义 K，可以按键字存取数据；如果不定义，则表示顺序读取。

DEVICE：DISK，数据文件必须固定定义。

```
FSCXXXA    IF   E            DISK
FSCXXXA    IF A E            DISK
FSCXXXA    O    E            DISK
FSCXXXL00  IF   E          K DISK
FSCXXXL00  UF   E          K DISK
FSCXXXL00  UF A E          K DISK
```

文件定义为 O 方式时，数据不是实时写到硬盘，而写入内存数据缓存区（系统默认为 4K），内存缓存满了才一次性地将缓存数据写入硬盘。如果缓存不满，程序会保留在内存中，直到文件关闭考会强制将数据写到硬盘。RPG 程序如果以 *INLR=*ON 结束，则包含了关闭文件操作，也会将数据写入硬盘。O 方式一般适用于写入的数据同时不会有其他应用访问。

因为在内存中的数据，其他程序是无法访问的，即可能出现程序 A 执行了写操作，程序 B 无法访问的情况。

如果需要满足程序 A 写入数据后，即调用程序 B，程序 B 需要使用程序 A 写入的数据。则文件的操作方式应该由 O 改为 I+A。

文件的 KEYWORDS 可以根据需要定义。每一行只定义一个 KEYWORDS，如果需要定义多个，则换行继续定义，提高系统的一致性、可读性。

5.1.2 相同格式名文件的定义方法

按开发规范的规定，物理文件、逻辑文件的格式名相同。如果在同一个程序中需要同时使用物理文件、逻辑文件或者相同物理文件的多个逻辑文件，则必须对文件的格式更名。

更新原则：物理文件的格式不更名，只对逻辑文件的格式名称进行更名。

更新方法：文件的格式名+逻辑文件序号

定义方法如下：

```
FTTXXXA    IF   E            DISK
FTTXXXL01  UF   E          K DISK    RENAME(RTTXXX:RTTXXX01)
FTTXXXL02  UF   E          K DISK    RENAME(RTTXXX:RTTXXX02)
```

按开发规范的规定，物理文件、逻辑文件的字段相同。如果在同一个程序中同时使用物理文件、逻辑文件或者相同物理文件的多个逻辑文件，则字段相同。读不同的文件时，字段内容会覆盖前一次其他文件的操作结果。如果这种情况会影响程序的逻辑和结果，则必须对字段名进行更名。

更新原则：物理文件的字段不更名，只对逻辑文件的字段名称进行更名。统一为一个文件的全部字段更名，不单独对某个字段更名。

更新方法：字段前缀+逻辑文件序号+字段名称。

定义方法如下：

```
FTTXXXA    IF   E            DISK
FTTXXXL01  UF   E          K DISK    RENAME(RTTXXX:RTTXXX01)
F                                    PREFIX(XXX01:3)
FTTXXXL02  UF   E          K DISK    RENAME(RTTXXX:RTTXXX02)
F                                    PREFIX(XXX02:3)
```

5.1.3 事务控制的定义方法

如果文件参与事务处理，则必须定义文件的 KEYWORDS 的 COMMIT。

定义方法：

```
FTTXXXL01  UF   E          K DISK    COMMIT
```

联机交易的事务控制由联机平台完成，一般组件程序不应该出现 COMMIT、ROLBAK 命令。

特殊情况下，可以定义文件为有条件事务处理，即可以选择文件是否定义 COMMIT。

定义原则：必须定义条件变量，变量名称为 C_CMTF_文件名称；文件的打开必须为 USROPN。

定义方法：

```
     FTTXXXL01  UF   E           K DISK    COMMIT(C_CMTF_TTXXXL01)
     F                                     USROPN
      *事务控制标志变量定义
     D C_CMTF_TTXXX01  S                 1
      /FREE

         // 文件不参与事务的方法
         C_CMTF_TTXXX01 = *OFF ;
         OPEN(E) TTXXXL01 ;
         CLOSE(E) TTXXXL01 ;

         // 文件参与事务的方法
         C_CMTF_TTXXX01 = *ON ;
         OPEN(E) TTXXXL01 ;
         CLOSE(E) TTXXXL01 ;

      /END-FREE
```

5.1.4 错误处理的定义方法

文件错误处理可以定义文件信息数据结构、文件处理子过程。

系统使用统一的文件信息数据结构：XDS_FILE，即一个程序中只对主数据文件采集文件信息。如果采集多个文件的信息，需要针对每个文件定义不同的数据结构名，数据结构的内容必须使用 LIKEDS 参考结构 XDS_FILE。数据结构的命名规则为：DS_F_文件名称。

系统统一使用*PSSR 做为文件操作异常处理子过程。如果有特殊要求，可以编写特定的处理子例程，子例程命名规则：@SR_文件名称，与一般子例程不同，名称可以到 12 位。

定义方法如下：

```
FTTXXXL01  UF   E           K DISK    INFDS(XDS_FILE)
F                                     INFSR(*PSSR)
F                                     RENAME(RTTXXX:RTTXXX01)
F                                     PREFIX(XXX01:3)
FTTXXXL02  UF   E           K DISK    INFDS(DS_F_TTXXXL02)
F                                     INFSR(@SR_TTXXXL02)
F                                     RENAME(RTTXXX:RTTXXX02)
F                                     PREFIX(XXX02:3)
```

5.1.5动态打开文件的定义方法

如果程序运行过程中需要处理结构相同的不同文件或成员，可以不再使用OVRDBF，直接使用关键字 EXTFILE、EXTMBR 即可。使用这种方式必须配合 USROPN。

如果只处理不同的文件，则单独使用 EXTFILE 即可。

EXTFILE 的使用方法：

EXTFILE()内定义常量，也可以定义变量；

变量的命名规则为：C_F_文件名称

变量类型为 21 位字符

变量内容：LIB 名称+’/’+文件名称　或　文件名称　或 ’*LIBL/’ + 文件名称。

EXTMBR 的使用方法：

EXTMBR()可以定义常量，也可以定义变量；

变量的命名规则：C_M_文件名称

变量类型为 10 位字符

变量内容：成员名称。

定义方法如下：

```
FTTXXXA    IF   E             DISK    EXTFILE(C_F_TTXXXA)
F                                     EXTMBR(C_M_TTXXXA)
F                                     USROPN
D C_F_TTXXXA      S               21
D C_M_TTXXXA      S               10
 /FREE
```

```
        C_F_TTXXXA = '*LIBL' + '/' + 'TTXXX12' ;
        C_M_TTXXXA = 'M201201' ;
        OPEN(E) TTXXXA ;
        CLOSE(E) TTXXXA ;
      /END-FREE
```

5.2 显示文件定义说明

一般显示文件定义方法：

```
     FTTXXXM2D  CF   E             WORKSTN INDDS(IND_DS)
```

子文件显示文件定义方法：

```
     FTTXXXM2D  CF   E             WORKSTN INDDS(IND_DS)
     F                                     SFILE(OPTSFL10:S_RECNUM10)
```

5.3 打印文件定义说明

定义方法：

```
     FTTXXXM90  O    E             PRINTER OFLIND(*IN90)
```

5.4 常用关键字的定义方法

USROPN 用于定义文件打开、关闭由程序控制。

BLOCK(*YES)用于定义文件按块处理，用于逻辑文件的定义。当适当的情况下可以减少 I/O 次数，提高处理性能。是否有效与数据的分布相关，使用时必须经过测试验证。以 U 方式使用文件时或有 READE、READP、READPE 操作时，系统忽略 BLOCK 的定义。

6 D 表定义方法

6.1 原型定义

6.1.1 交易组件原型定义

自由格式 RPG 不允许使用*ENTRY 接收程序参数。必须使用程序原型定义接收程序参数。

定义方式如下：

```
     *****************************************************************
     *组件调用接口定义
     *****************************************************************
     D zznnnnR          PR                  EXTPGM('zznnnnR')
     D  PR_TWA                              LIKEDS(XDS_SCCTWA)
     D  PR_zznnnnI                          LIKEDS(zznnnnI)
     D zznnnnR          PI
     D  PI_TWA                              LIKEDS(XDS_SCCTWA)
     D  PI_zznnnnI                          LIKEDS(zznnnnI)
```

定义分为两部分：

1. D……PR 程序定义：

   zznnnnR 与组件程序名称相同；

   PR 固定表程序定义；

   EXTPGM（）中定义实际的程序名称，应该当前的程序名称相同；

   参数个数及类型定义，

   PR_表示程序参数定义，这里定义的参数只是为了说明参数的个数、顺序、类型，不是实际参数，PR_不能做为参数在程序中使用；

   参数类型定义尽量使用 LIKE 关键字参考已经定义的数据结构、变量。

2. D……PI 程序参数定义：

   zznnnnR 与组件程序名称相同；

   PI 固定表程序接口定义；

   参数一：PI_TWA，使用 LIKEDS 参考 XDS_SCCTWA 定义

   参数二：PI_zznnnnI，组件输入接口，zznnnnI 与组件输入接口名称相

同，使用 LIKEDS(zznnnnI)定义。

6.1.2其他服务组件的原型定义

其他服务组件的原型定义与交易组件的原型定义方法类似。

组件的接口参数固定为两个，第一个固定为 TWA、第二个为组件接口数据结构。

组件的原型定义按以下格式定义：

```
     ****************************************************************
     *组件调用接口定义
     ****************************************************************
     D zzxxxxxxR       PR                  EXTPGM('zzxxxxxxR')
     D  PR_TWA                             LIKEDS(XDS_SCCTWA)
     D  PR_zzxxxxxx                        LIKEDS(zzxxxxxx)
     D zzxxxxxxR       PI
     D  PI_TWA                             LIKEDS(XDS_SCCTWA)
     D  PI_zzxxxxxx                        LIKEDS(zzxxxxxx)
```

定义分为两部分：

1. D……PR 程序定义：

   zzxxxxxxR 与组件程序名称相同；

   PR 固定表程序定义；

   EXTPGM（）中定义实际的组件程序名称，应该当前的程序名称相同；

   两个参数定义：

   第一个参数固定为 PR_TWA，必须使用 LIKEDS（XDS_SCCTWA）方式定义；

   第二个参数固定为组件接口，定义为 PR_zzxxxxxx，zz 为应用模块标识、xxxxxx 为组件名称；

2. D……PI 程序参数定义：

   zzxxxxxxR 与组件程序名称相同；

   PI 固定表程序接口定义；

   两个参数定义：

   第一个参数固定为 PI_TWA，必须使用 LIKEDS（XDS_SCCTWA）方式定义；

   第二个参数固定为组件接口，定义为 PI_zzxxxxxx，zz 为应用模块标识、xxxxxx 为组件名称；

6.1.3 单元组件、函数组件的原型定义

单元组件、函数组件与其他服务组件的原型定义方法类似。

组件的接口参数固定为两个，第一个固定为 TWA、第二个为组件接口数据结构。

组件的原型定义按以下格式定义：

```
     *****************************************************************
     *组件调用接口定义
     *****************************************************************
     D zzxxxxxxR       PR                  EXTPGM('zzxxxxxxR')
     D  PR_TWA                             LIKEDS(XDS_SCCTWA)
     D  PR_zzxxxxxx                        LIKEDS(zzxxxxxx)
     D zzxxxxxxR       PI
     D  PI_TWA                             LIKEDS(XDS_SCCTWA)
     D  PI_zzxxxxxx                        LIKEDS(zzxxxxxx)
```

定义分为两部分：

3. D……PR 程序定义：

   zzxxxxxxR 与组件程序名称相同；

   PR 固定表程序定义；

   EXTPGM（）中定义实际的组件程序名称，应该当前的程序名称相同；

   两个参数定义：

   第一个参数固定为 PR_TWA，必须使用 LIKEDS（XDS_SCCTWA）方式定义；

   第二个参数固定为组件接口，定义为 PR_zzxxxxxx，zz 为应用模块标识、xxxxxx 为组件名称；

4. D……PI 程序参数定义：

   zzxxxxxxR 与组件程序名称相同；

   PI 固定表程序接口定义；

   两个参数定义：

   第一个参数固定为 PI_TWA，必须使用 LIKEDS（XDS_SCCTWA）方式定义；

   第二个参数固定为组件接口，定义为 PI_zzxxxxxx，zz 为应用模块标识、xxxxxx 为组件名称；

6.2 数据结构定义

6.2.1 参考外部文件定义数据结构

在程序中参考文件结构定义数据结构。

一般情况下，参考文件结构定义数据结构时，应该参考物理文件结构进行定义，数据结构的命名方式：DS_zzfff，其中：

DS_：固定表示数据结构；

zz：应用模块标识；

fff：文件标识；

zzfff 与对应文件名称的前五位相同 yyyyy 为文件的名称。

如果应用必须参考逻辑文件定义结构时，数据结构的命名方式：DD_zzfffnn，其中：

DS_：固定表示数据结构；

zz：应用模块标识；

fff：文件标识；

nn：逻辑文件序号；

zzfff 与对应文件名称的前五位相同，nn 与文件名称的后两位相同。

如果不同的数据结构中有同名字段，则定义数据结构时必须使用关键字 QUALIFIED。使用数据结构的字段时，不能只字段名称，必须在字段名称前+‘数据结构名.’做为字段的前缀。程序将字段名称、数据结构名.字段名称做为不同的变量使用。

定义相同结构的数据结构时，可以参考同一个文件定义，但只能有一个数据结构不使用 QUALIFIED，其他都必须使用 QUALIFIED。可以使用 LIKEDS 定义新的数据结构参考已经定义的数据结构，这两个结构含有相同的字段数、字段名称、字段类型。这种方式隐含了 QUALIFIED。

结构相同的数据结构命名时，在基本数据结构的名称+’_c’，’_’用于分隔，c 用于标识数据结构。

例：定义保存文件修改前、后数据的数据结构，文件的数据结构为 DS_zzfff，改前数据结构名为 DS_zzfff_B、改后数据结构名为 DS_zzfff_A。

定义数据结构时，多个关键字必须分行定义，即每行只能定义一个关键字（KEYWORDS）。

在程序中，可以把数据结构做为一个变量使用，也可以单独使用数据结构内的字段。

可以为整个数据结构赋值，也可以为数据结构内的字段赋值。

可以使用 EVAL-CORR 为两个数据结构中同名字段赋值。

使用 CLEAR、RESET 命令对数据结构的字段做初始化处理。

定义及使用方法：

```
  *文件 TTXXXA 数据结构
 D DS_TTXXX       E DS                    EXTNAME(TTXXXA)
 D DS_TTXXX_A     E DS                    EXTNAME(TTXXXA)
 D                                        QUALIFIED
 D DS_TTXXX_B       DS                    LIKEDS(DS_TTXXX)

  /FREE
     DS_TTXXX_B = DS_TTXXX ;
     DS_TTTXX_A.fffnnnn = fffnnn ;
     CLEAR DS_TTXXX ;
     DS_TTXX = DS_TTXXX_B ;
     EVAL-CORR DS_TTXXX_B = DS_TTXXX_A ;
     RESET DS_TTXXX ;

  /END-FREE
```

6.2.2定义文件键字结构

在自由格式中不允许 KLIST 定义键字列表，而是直接参考文件结构定义键字数据结构。

在程序中使用%KDS()代替键字列表的操作。

使用%KDS 时可以根据需要选择使用键字数据结构的字段数，不需要像 KLIST 为不同的键字数量定义多个键字列表。

键字数据结构的名称方式：DS_zzfff_Knn，其中：

DS_：固定表示数据结构

zz：应用模块标识；

fff：为文件名称；

K：表示键字结构；

nn：为逻辑文件序号；

zzfff 与对应文件的前五位相同，nn 与对应文件的后两位相同。

为了区分键字段与文件字段，以区分不同逻辑文件的键字段，必须使用 PREFIX 为键字结构的字段定义前缀，前缀为 Knn_，其中：

K：固定表示键字；

nn：为逻辑文件序号，与对应文件的后两位相同；

_：用于分隔；

由于键字结构中的字段名称不同，一般不再需要使用关键字 QUALIFIED。

定义数据结构时，多个关键字必须分行定义，即每行只能定义一个关键字（KEYWORDS）。

定义及使用方法：

```
  *文件 TTXXXL01 定义
FTTXXXL01  UF   E             DISK


  *文件 TTXXXL01 键字结构
```

```
D DS_TTXXX_K01   E DS                    EXTNAME(TTXXXL01:*KEY)
D                                        PREFIX(K01_)


 /FREE
    K01_XXXTYPE = 'AAAAA' ;
    K01_XXXCODE = *LOVAL ;
    //循环取等于第一个键字数据
    SETLL %KDS(DS_TTXXX_K01:1) TTXXXL01 ;
    IF %EQUAL() ;
       READE %KDS(DS_TTXXX_K01:1) TTXXXL01 ;
       DOW NOT %EOF ;
          //处理流程
          READE %KDS(DS_TTXXX_K01:1) TTXXXL01 ;
       ENDDO ;
    ENDIF ;



    //取等于全部键字数据
    CHAIN(E) %KDS(DS_TTXXX_K01) TTXXXL01 ;
    IF NOT %ERROR AND %FOUND ;
       //处理流程
    ENDIF ;


 /END-FREE
```

6.2.3程序用数据结构定义

数据结构的名称：DS_xxxxxx，其中：

DS_：表示数据结构定义；

xxxxxx：结构名称。

数据结构的字段名称：z_yyyy，其中：

z：为字段类型；

yyyy：为字段名称。

z 字段类型分为：

B 二进制型，对应 B 型，一般 2 个字节的二进制数定义为 5I0,4 个字节的二进制数定义为 10I0；

C 字符型，对应 A/O 型；

P 压缩数字，对应 P 型；

S 区位数字，对应 S 型；

D 日期型，对应 D 型；

T 时间型，对应 T 型；

Z 时间戳，对应 Z 型；

I 指示器，对应 N 型，逻辑型；

定义字段时应该使用 LIKE 参考数据库字段或已经定义的变量，字段名称应该与被参考的字段名称相同。

如果没有对应的数据字典或参考字段，则可以直接定义字段的类型、长度、小数位。

为了避免与其他数据结构的字段相同，应用关键字 QUALIFIED。

定义字段时只需要定义字段的长度，由系统自动计算数据结构的长度。不要使用定义字段起始位置、结束位置的方式。定义数据结构与系统数据结构对应时，可以使用字段起始位置、结束位置的方式。

```
     *数据结构说明
    D DS_TTYYY          DS                    QUALIFIED
    D   C_CODE                                LIKE(XXXCODE)
```

```
     D  C_TYPE                          LIKE(XXXTYPE)
     D  D_DATE                     D    DATFMT(*ISO)
     D  T_TIME                     T    TIMFMT(*ISO)
```

6.2.4交易服务组件接口数据结构的定义

6.2.4.1 组件输入接口数据结构的定义

组件输入接口数据结构的名称统一为zznnnnI。

先使用 E+DS+EXTNAME 参考交易服务组件接口文件分别定义数据结构zznnnnIS、zznnnnIM，再定义数据结构zznnnnI，此数据结构基于指针。例：

```
* 组件输入接口定义
D zznnnnIS       E DS                  EXTNAME(zznnnnIS)
D                                      QUALIFIED
D                                      BASED(PTR_zznnnnIS)
D zznnnnIM       E DS                  EXTNAME(zznnnnIM)
D                                      QUALIFIED
D                                      BASED(PTR_zznnnnIS)
D                                      DIM(10)
D zznnnnI          DS                  QUALIFIED
D                                      BASED(PTR_zznnnnI)
D  IS                                  LIKEDS(zznnnnIS)
D  IM                                  LIKEDS(zznnnnIM)
```

结构结构中：

zz：应用模块标识；

nnnn：交易码。

6.2.4.2 组件输出接口数据结构的定义

组件输出接口数据结构的名称统一为zznnnnE。

先使用 E+DS+EXTNAME 参考交易服务组件接口文件分别定义数据结构zznnnnES、zznnnnEM，再定义数据结构zznnnnE，此数据结构基于指针。例：

```
* 组件输出接口定义
D zznnnnES       E DS                  EXTNAME(zznnnnES)
D                                      QUALIFIED
D                                      BASED(PTR_zznnnnES)
D zznnnnEM       E DS                  EXTNAME(zznnnnEM)
```

```
D                                       QUALIFIED
D                                       BASED(PTR_zznnnnEM)
D                                       DIM(10)
D zznnnnE           DS                  QUALIFIED
D                                       BASED(PTR_zznnnnE)
D  ES                                   LIKEDS(zznnnnES)
D  EM                                   LIKEDS(zznnnnEM)
```

结构结构中：

zz：应用模块标识；

nnnn：交易码。

6.2.4.3 组件多页查询输出数据结构的定义

组件多页查询输出数据结构的名称统一为 zznnnnL，直接使用 E+DS+EXTNAME 参考交易服务组件接口文件定义数据结构 zznnnnL。例：

```
* 多页查询输出接口定义
D zznnnnEL        E DS                  EXTNAME(zznnnnEL)
D                                       QUALIFIED
```

6.2.5 其他服务组件接口数据结构的定义

其他服务组件接口数据结构的名称统一为 zzxxxxxx。

不含循环字段的接口数据结构定义，则需要先使用 E+DS+EXTNAME 参考服务组件接口文件分别定义组件输入数据结构 zzxxxxxxIS、组件输出数据结构 zzxxxxxxES，再定义组件接口数据结构 zzxxxxxx，此数据结构基于指针。例：

```
* 组件接口定义
D zzxxxxxxIS      E DS                  EXTNAME(zzxxxxxxIS)
D                                       QUALIFIED
D                                       BASED(PTR_zzxxxxxxIS)
D zzxxxxxxES      E DS                  EXTNAME(zzxxxxxxES)
D                                       QUALIFIED
D                                       BASED(PTR_zzxxxxxxES)
D zzxxxxxx          DS                  QUALIFIED
D                                       BASED(PTR_zzxxxxxx)
D  IS                                   LIKEDS(zzxxxxxxIS)
D  ES                                   LIKEDS(zzxxxxxxES)
```

包含循环字段的接口数据结构定义，则需要先使用 E+DS+EXTNAME 参考服务组件接口文件分别定义组件输入接口数据结构 zzxxxxxxIS、zzxxxxxxIM、定义组件输出接口数据结构 zzxxxxxxES、zzxxxxxxEM，再定义组件接口数据结构

zzxxxxxx，此数据结构基于指针。例：

```
* 组件接口定义
D zzxxxxxxIS     E DS                  EXTNAME(zzxxxxxxIS)
D                                      QUALIFIED
D                                      BASED(PTR_zzxxxxxxIS)
D zzxxxxxxIM     E DS                  EXTNAME(zzxxxxxxIM)
D                                      QUALIFIED
D                                      BASED(PTR_zzxxxxxxIM)
D                                      DIM(10)
D zzxxxxxxES     E DS                  EXTNAME(zzxxxxxxES)
D                                      QUALIFIED
D                                      BASED(PTR_zzxxxxxxES)
D zzxxxxxxEM     E DS                  EXTNAME(zzxxxxxxEM)
D                                      QUALIFIED
D                                      BASED(PTR_zzxxxxxxEM)
D                                      DIM(10)
D zzxxxxxx         DS                  QUALIFIED
D                                      BASED(PTR_zzxxxxxx)
D  IS                                  LIKEDS(zzxxxxxxIS)
D  IM                                  LIKEDS(zzxxxxxxIM)
D  ES                                  LIKEDS(zzxxxxxxES)
D  EM                                  LIKEDS(zzxxxxxxEM)
```

只包含循环字段的接口数据结构定义，则需要先使用 E+DS+EXTNAME 参考服务组件接口文件分别定义组件输入接口数据结构 zzxxxxxxIM、定义组件输出接口数据结构 zzxxxxxxOM，再定义组件接口数据结构 zzxxxxxx，此数据结构基于指针。例：

```
* 组件接口定义
D zzxxxxxxIM     E DS                  EXTNAME(zzxxxxxxIM)
D                                      QUALIFIED
D                                      BASED(PTR_zzxxxxxxIM)
D                                      DIM(10)
D zzxxxxxxEM     E DS                  EXTNAME(zzxxxxxxEM)
D                                      QUALIFIED
D                                      BASED(PTR_zzxxxxxxEM)
D                                      DIM(10)
D zzxxxxxx         DS                  QUALIFIED
D                                      BASED(PTR_zzxxxxxx)
D  IM                                  LIKEDS(zzxxxxxxIM)
D  EM                                  LIKEDS(zzxxxxxxEM)
```

结构结构中：

zz：应用模块标识；

xxxxxx：组件名称。

6.2.6单元组件、函数组件接口数据结构的定义

6.2.6.1 定义方法

单元组件、函数组件接口数据结构的名称统一为zzxxxxxx。

不含循环字段的接口数据结构定义，则需要先使用 E+DS+EXTNAME 参考组件接口文件分别定义组件输入数据结构 zzxxxxxxIS、组件输出数据结构zzxxxxxxES，再定义组件接口数据结构zzxxxxxx，此数据结构基于指针。例：

```
* 组件接口定义
D zzxxxxxxIS     E DS                  EXTNAME(zzxxxxxxIS)
D                                      QUALIFIED
D                                      BASED(PTR_zzxxxxxxIS)
D zzxxxxxxES     E DS                  EXTNAME(zzxxxxxxES)
D                                      QUALIFIED
D                                      BASED(PTR_zzxxxxxxES)
D zzxxxxxx         DS                  QUALIFIED
D                                      BASED(PTR_zzxxxxxx)
D  IS                                  LIKEDS(zzxxxxxxIS)
D  ES                                  LIKEDS(zzxxxxxxES)
```

包含循环字段的接口数据结构定义，则需要先使用 E+DS+EXTNAME 参考组件接口文件分别定义组件输入接口数据结构 zzxxxxxxIS、zzxxxxxxIM、定义组件输出接口数据结构 zzxxxxxxES、zzxxxxxxEM，再定义组件接口数据结构zzxxxxxx，此数据结构基于指针。例：

```
* 组件接口定义
D zzxxxxxxIS     E DS                  EXTNAME(zzxxxxxxIS)
D                                      QUALIFIED
D                                      BASED(PTR_zzxxxxxxIS)
D zzxxxxxxIM     E DS                  EXTNAME(zzxxxxxxIM)
D                                      QUALIFIED
D                                      BASED(PTR_zzxxxxxxIM)
D                                      DIM(10)
D zzxxxxxxES     E DS                  EXTNAME(zzxxxxxxES)
D                                      QUALIFIED
D                                      BASED(PTR_zzxxxxxxES)
D zzxxxxxxEM     E DS                  EXTNAME(zzxxxxxxEM)
D                                      QUALIFIED
```

```
D                                       BASED(PTR_zzxxxxxxEM)
D                                       DIM(10)
D zzxxxxxx          DS                  QUALIFIED
D                                       BASED(PTR_zzxxxxxx)
D  IS                                   LIKEDS(zzxxxxxxIS)
D  IM                                   LIKEDS(zzxxxxxxIM)
D  ES                                   LIKEDS(zzxxxxxxES)
D  EM                                   LIKEDS(zzxxxxxxEM)
```

只包含循环字段的接口数据结构定义，则需要先使用 E+DS+EXTNAME 参考组件接口文件分别定义组件输入接口数据结构 zzxxxxxxIM、定义组件输出接口数据结构 zzxxxxxxEM，再定义组件接口数据结构 zzxxxxxx，此数据结构基于指针。

例：

```
* 组件接口定义
D zzxxxxxxIM     E DS                   EXTNAME(zzxxxxxxIM)
D                                       QUALIFIED
D                                       BASED(PTR_zzxxxxxxIM)
D                                       DIM(10)
D zzxxxxxxEM     E DS                   EXTNAME(zzxxxxxxEM)
D                                       QUALIFIED
D                                       BASED(PTR_zzxxxxxxEM)
D                                       DIM(10)
D zzxxxxxx          DS                  QUALIFIED
D                                       BASED(PTR_zzxxxxxx)
D  IM                                   LIKEDS(zzxxxxxxIM)
D  EM                                   LIKEDS(zzxxxxxxEM)
```

结构结构中：

zz：应用模块标识；

xxxxxx：组件名称。

6.2.6.2 使用方法

由于服务组件、单元组件会被服务组件及其他组件调用，为了避免每个使用的组件都定义组件调用方法、组件接口，应该将服务组件、单元组件的原型定义、接口定义做为公用定义，定义保存在源文件 CzzDFNSRC，源成员名称为 zzxxxxxxH（与组件程序名称的前 8 位相同）。

服务组件程序、单元组件程序及调用这些组件的程序都不需要定义组件原型、接口，使用/COPY 引用公用定义。

6.3 程序变量定义

6.3.1 程序变量定义

在自由格式 RPG 中不允许在 C 中定义程序变量，必须在 D 表中定义程序变量。

变量名称：z_yyyyt，其中：

z:字段类型；

yyyy：字段名称；

t：字段后缀，当一个程序中需要多个相同属性的字段，可以用后缀加以区分，后缀可以使用字母或数字。例程序中需要代码字段，同时又需要定义代码的处理范围，即代码的上、下限，就可以三个字段：C_CODE 用于处理、C_CODEL 表示代码的下限、C_CODEH 表示代码的上限。

字段类型分为：

B 二进制型，对应 B 型，一般 2 个字节的二进制数定义为 5I0,4 个字节的二进制数定义为 10I0；

C 字符型，对应 A/O 型；

P 压缩数字，对应 P 型；

S 区位数字，对应 S 型；

D 日期型，对应 D 型；

T 时间型，对应 T 型；

Z 时间戳，对应 Z 型；

I 指示器，对应 N 型，逻辑型；

定义变量时可以采用 LIKE 参考数据库字段或已经定义的变量，尽量采用这种方式，数据结构的字段名称应该与被参考的字段名称相同。也可以直接定义变量的类型、长度、小数位。

程序变量应该集中定义，在变量定义前必须增加变量定义代码段的注释。不能直接从变量名称理解含义的变量，必须在变量定义的上一行增加注释。不同类型的变量定义间应该保留一行空行。

变量定义时应该按类型的字母顺序定义，再按字段名称的字母顺序定义，以方便查找。

定义方法如下：

```
     ****************************************************************
     *变量定义
     ****************************************************************
     *代码
    D C_CODE          S                   LIKE(XXXCODE)
     *代码上限
    D C_CODEH         S                   LIKE(XXXCODE)
     *代码下限
    D C_CODEL         S                   LIKE(XXXCODE)
     *系统日期
    D D_DATE          S               D   DATFMT(*ISO)
     *系统时间
    D T_TIME          S               T   TIMFMT(*ISO)
```

6.3.2程序常量定义

常量名称：L_xxxx_yyyy，其中：

L_：表示程序定义的常量；

xxxx：为常量类别，一般与对应的数据字典相同；

yyyy：为常量取值；

程序常量应该集中定义，在常量定义前必须增加常量定义代码段的注释。不能直接从常量定义理解含义的常量，必须在常量定义的上一行增加注释。

如果一个常量类别只定义一个常量值，则不需要与其他常量定义间保留空行；如果一个常量类别定义多个常量值，则每一组常量定义间保留一行空行。

常量定义时应该按常量类别的字母顺序定义，再按常量取值的字母顺序定义，以方便查找。

```
     ******************************************************************
     *常量定义
     ******************************************************************
     *日志文件在线标志
     *                                   ---不在线---
    D L_ONLF_N         C                CONST('0')
     *                                   ---在线---
    D L_ONLF_Y         C                CONST('1')

     *日志文件保存标志
     *                                   ---不保存---
    D L_SAVF_N         C                CONST('0')
     *                                   ---保存---
    D L_SAVF_Y         C                CONST('1')
```

6.3.3数组定义

数组名称：A_yyyy，其中：

A：固定表示数组；

yyyy：数组名称，如果参考数据字典或字段定义，则数组名称应该与字段名称相同；

数组定义时应该按数组名称的字母顺序定义，以方便查找。

定义方法如下：

```
     ****************************************************************
     *数组定义
     ****************************************************************
     *代码数组
D A_CODE          S                   LIKE(XXXCODE)
D                                     DIM(10)
```

6.3.4 指针型数据定义

指针名称：PTR_yyyyyy，其中：

PTR_：固定表示指针；

yyyyyy：指针名称，可以为 3—6 个字符，如果指针指向数据结构，则指针名称与数据结构名称相同；如果指针程序变量，则则指针名称与字段名称相同。

指针变量可以单独定义，也可以使用关键字 BASED 定义。系统默认 BASED()中的变量为指针，不需要再单独定义。

指针定义时应该按指针名称的字母顺序定义，以方便查找。

定义方法如下：

```
     *TWA 指针
D PTR_TWA         S                   *


     *TIA 数据结构，基于指针 PTR_TIA
D DS_SCCTIA       E DS                *   LIKEDS(XDS_SCCTIA)
D                                         BASED(PTR_TIA)
     *TOA 数据结构，基于指针 PTR_TOA
D DS_SCCTOA       E DS                *   LIKEDS(XDS_SCCTOA)
D                                         BASED(PTR_TOA)
```

6.4 公用数据定义

6.4.1 公用程序原型定义

公用程序原型定义按应用模块划分，如果模块比较大可以再进一步按功能。

公用程序原型定义源码统一保存在源文件 C@@DFNSRC 中。

公用程序原型定义源成员名称：zzPUBDFNnH，其中：

zz：应用模块标识；

PUBDFN：表示公用程序原型定义（包含公用过程原型定义）

n：为定义序号，由应用模块划分使用。

H：表示定义源码（不能进行编译）

公用程序原型定义与一般程序原型定义方法相同，只是在 PGM_或 PROC_之前增加字母 X，即以 XPGM_表示公用程序、XPROC_表示公用过程。

在程序原型定义前，需要增加公用程序的描述，包括：程序名称、功能描述、入口参数、返回参数等信息。

为了防止程序引用时出现重复定义，在源成员说明注释之后，应该增加/DEFINED 的定义说明：

```
/IF NOT DEFINED(源成员名称)
/DEFINE 源成员名称
```

在程序之前，增加定义结束：

```
/ENDIF
```

定义方法如下：

```
/IF NOT DEFINED(SCPUBDFN2H)
/DEFINE SCPUBDFN2H

*****************************************************************
*  程序名称：  XPGM_SDMSG
*  功能描述：抛信息处理
```

```
     *  接口参数:
     *       01    *          TWA 接口
     *       02    *          SDMSG 接口
     *  返回参数:
     *     无
     ******************************************************************
    D XPGM_SCSDMSG     PR                  EXTPGM('SCSDMSGB')
    D  PR_TWA                              LIKE(XDS_SCCTWA)
    D  PR_SDMSG                            LIKE(XDS_SCCSDMSG)

     ******************************************************************
     *  程序名称:  XPGM_CLCPN
     *  功能描述:调用组件
     *  接口参数:
     *       01    *          TWA 接口
     *       02    *          CLCPN 接口
     *  返回参数:
     *     无
     ******************************************************************
    D XPGM_SCCLCPN     PR                  EXTPGM('SCCLCPNB')
    D  PR_TWA                              LIKE(XDS_SCCTWA)
    D  PR_CLCPN                            LIKE(XDS_SCCCLCPN)

     /ENDIF
```

技术平台提供的公用程序原型定义:

SCOLPDFN2H:抛信息、组件调用、取参数、格式输出、跟踪调试等程序原型定义;

SCOLPDFN3H;例外处理程序原型定义。

6.4.2公用数据结构定义

公用数据结构定义按应用模块划分,如果模块比较大可以再进一步按功能。

公用数据结构定义源码统一保存在源文件 C@@DFNSRC 中。

公用数据结构定义源成员名称:zzDTADFNnH,其中:

zz:应用模块标识;

DTADFN:表示公用变量定义

n:为定义序号,由应用模块划分使用。

全系统公用的数据结构定义可以定义为@@DTADFNnH 中。

公用数据结构的命名方式

公用数据结构命名方式与一般程序数据结构的命名方式基本相同，只是在数据结构名称的‘DS_’前增加字母 X，表示是公用数据结构，即公用数据结构的命名方式：XDS_xxxxxx。

应用程序应该根据需要，参考公用数据结构定义程序使用的数据结构，而不是直接使用公用数据结构。所以公用数据结构不应该占用变量内存空间，为避免公用数据结构占用内存，在定义公用数据结构时应用使用关键字 BASED(@)，实现即不占用内存，又防止应用直接使用。

为了防止程序引用时出现重复定义，在源成员说明注释之后，应该增加/DEFINED 的定义说明：

```
/IF NOT DEFINED(源成员名称)
/DEFINE 源成员名称
```

在程序之前，增加定义结束：

```
/ENDIF
```

定义方法如下：

```
/IF NOT DEFINED(SCDTADFN1H)
/DEFINE SCDTADFN1H

*****************************************************************
* 数据结构定义
*****************************************************************
* 系统输入区结构
D XDS_SCCTIA     E DS                  EXTNAME(SCTIAA)
D                                      QUALIFIED
D                                      BASED(@)
* 系统输出区结构
D XDS_SCCTOA     E DS                  EXTNAME(SCTOAA)
```

```
     D                                       QUALIFIED
     D                                       BASED(@)

      /ENDIF
```

技术平台提供的公用定义，包括以下内容：

SCDTADFN1H：TIA、TOA、SCA 等技术平台数据区结构定义；

SCDTADFN2H：TWA、抛信息组件接口、组件调用组件接口、取参数组件接口、格式输出组件接口、跟踪调试组件接口；

SCDTADFN3H：例外处理组件接口。

系统公用的数据结构定义@@DTADFN1H，包括：程序反馈信息结构、文件反馈信息结构。

内容如下：

```
     ****************************************************************
     ****************************************************************
     ** (C) Copyright  -  中信银行  CHINA CITIC BANK                *
     **                  Year 2005, all rights reserved.           **
     ****************************************************************
     ****************************************************************

     **BBB***********************************************************
     *                                                               *
     *   系统名称 : 中信银行核心系统                                 *
     *   程序名称 : @@DTADFN1H                                       *
     *   功能描述 : 系统公用数据定义一                               *
     *   程序员名 : 开发人员                                         *
     *   编程日期 : 2012/04/14                                       *
     *                                                               *
     **EEE***********************************************************
```

```
/IF NOT DEFINED(@@DTADFN1H)
/DEFINE @@DTADFN1H

*****************************************************************
*数据结构定义
*****************************************************************
*程序反馈信息结构
D                 SDS
D XC_SYPN           *PROC
D XS_SYST           *STATUS
D XC_PSTS             16     20   0
D XC_LINE             21     28
D XC_MSGT             40     42
D XC_MSGN             43     46
D XC_SYPL             81     90
D XC_FILE            175    184
D XC_STSI            209    243
D XC_SYJN            244    253
D XC_SYUI            254    263
D XC_SYJO            264    269
D XC_SRCF            304    313
D XC_SRCL            314    323
D XC_SRCM            324    333
D XC_PPRC            334    343
D XC_MPRC            344    353
D XB_STN1            354    355I 0
D XB_STN2            356    357I 0
D XC_CURU            358    367
```

```
     *文件反馈信息结构
     D XDS_FILE        DS                  QUALIFIED
     D XC_FIL8                1       8
     D XC_OPNF                9       9
     D XC_ENDF               10      10
     D XS_SYST               11      15  0
     D XC_OPCD               16      21
     D XC_ROUT               22      29
     D XC_LINE               30      37
     D XC_RECD               38      45
     D XB_STN1               77      78I 0
     D XB_FBSZ              367     370I 0
     D XB_LCKN              377     378I 0
     D XB_KEYN              387     388I 0
     D XB_KEYL              393     394I 0
     D XB_MBRN              395     396I 0
     D XB_DBRN              397     400I 0
     D XC_KEYC              401     500


      /ENDIF
```

6.4.3 公用变量定义

公用变量定义按应用模块划分，如果模块比较大可以再进一步按功能。

公用变量定义源码统一保存在源文件 C@@DFNSRC 中。

公用变量定义源成员名称：zzDTADFNnH，其中：

zz：应用模块标识；

DTADFN：表示公用变量定义

n：为定义序号，由应用模块划分使用。

公用变量与公用数据结构可以定义在同一个源文件中。

公用变量命名方式与一般程序变量的命名方式基本相同，只是在字段类型前增加字母 X，表示是公用变量，即公用变量命名规则：Xz_yyyy。

定义变量应该使用关键字 LIKE 参考数据字典、数据库字段或已经定义的变量，数据结构的字段名称应该与被参考的字段名称相同。

如果没有可参考字段，也可以直接定义变量的类型、长度、小数位。

变量定义时应该按类型的字母顺序定义，再按字段名称的字母顺序定义，以方便查找。

为了防止程序引用时出现重复定义，在源成员说明注释之后，应该增加/DEFINED 的定义说明：

```
/IF NOT DEFINED(源成员名称)
/DEFINE 源成员名称
```

在程序之前，增加定义结束：

```
/ENDIF
```

定义方法如下：

```
/IF NOT DEFINED(SCDTADFN1H)
/DEFINE SCCSTDFN1H

*信息代码
D XC_MSCD         S                   LIKE(@@MSCD)
*信息类型
D XC_MSGT         S                   LIKE(@@MSGT)
```

```
/ENDIF
```

6.4.4 公用常量定义

公用常量定义按应用模块划分，如果模块比较大可以再进一步按功能。

公用常量定义源码统一保存在源文件 C@@DFNSRC 中。

公用常量定义源成员名称：zzCSTDFNnH，其中：

zz：应用模块标识；

CSTDFN：表示公用常量定义

n：为定义序号，由应用模块划分使用。

公用常量命名方式与一般程序常量的命名方式基本相同，只是在将首字母 L 换为 G，表示是公用变量，即公用常量命名方式：G_xxxx_yyyy。

常量定义时应该按常量类别的字母顺序定义，再按常量取值的字母顺序定义，以方便查找。

为了防止程序引用时出现重复定义，在源成员说明注释之后，应该增加 /DEFINED 的定义说明：

```
/IF NOT DEFINED(源成员名称)
/DEFINE 源成员名称
```

在程序之前，增加定义结束：

```
/ENDIF
```

定义方法如下：

```
/IF NOT DEFINED(SCCSTDFN1H)
/DEFINE SCCSTDFN1H


*信息类型定义
```

```
     *                                   ---授权信息---
D G_MSGT_U          C                   CONST('A')
     *                                   ---提示信息---
D G_MSGT_P          C                   CONST('P')
     *                                   ---错误信息---
D G_MSGT_R          C                   CONST('R')
     *                                   ---警告信息---
D G_MSGT_W          C                   CONST('W')


     /ENDIF
```

6.4.5公用定义的使用

D表的定义顺序：

程序原型定义

调用程序原型定义

数据结构定义

键字结构定义

程序变量定义

程序常量定义

/COPY引用公用定义

应用程序中使用/COPY命令引用公用定义。

在/COPY命令中只能出现源文件名称、源成员名称，不允许出现源码库名称。

/COPY应该集中编写，按源文件、源成员名称顺序编写。

在/COPY代码前增加注释。

在程序可以参考公用定义定义程序变量。

RPG 程序不要求/COPY 公用定义一定在定义程序变量之前。所以/COPY 放在 D 表的最后，即在程序常量定义之后。

方法如下：

```
D DS_SCCTWA        DS                  LIKEDS(XDS_SCCTWA)
D DS_SCCSDMSG      DS                  LIKEDS(XDS_SCCSDMSG)


*****************************************************************
*公用定义引用
*****************************************************************
/COPY C@@DFNSRC,SCDTADFN1H
/COPY C@@DFNSRC,SCDTADFN2H
/COPY C@@DFNSRC,SCPUBSRV2H


/FREE
    DS_SCCSDMSG.SDMSGMSCD = ‘SCR0001’;
    F_SCSDMSG(PTR_TWA:’SCR0001’:*OMIT:*MOIT:*OMIT:*OMIT) ;
/END-FREE
```

7 C表定义方法

7.1 程序的基本编写方法

```
     /FREE

       C_MSGT = G_MSGT_E ;
       C_MSCD = 'SCR0001' ;

       IF A < B ;
          A = A + 1 ;
       ELSE ;
          IF B < A ;
             B = B + 1 ;
          ENDIF ;
       ENDIF ;

       SETLL %KDS(DS_TTXXX_K00) TTXXXL00 ;
       READE %KDS(DS_TTXXX_K00) TTXXXL00 ;
       DOW NOT %EOF ;
         // 处理流程
         READE %KDS(DS_TTXXX_K00) TTXXXL00 ;
      ENDDO ;

      SELECT ;
         WHEN A = 'C' ;
            EXSR @SR_CHGDTA
         WHEN A = 'D' ;
            EXSR @SR_DLTDTA
         OTHER ;
            EXSR @SR_DSPDTA
      ENDSL ;

      *INLR = *ON ;
      RETURN ;

       // ----------------------------------------------------
       //  数据修改子例程
       // ----------------------------------------------------
       BEGSR @SR_CHGDTA ;
```

```
            CHAIN %KDS(DS_TTXXX_K00) TTXXXL00 ;
            IF %FOUND ;
               XXXCODE = 'AAAA' ;
               UPDATE RTTXXX ;
            ENDIF ;

         ENDSR ;

      ENDSR


   /END-FREE
```

程序以/FREE 开始、以/END-FREE 结束。第一行源码与/FREE 保留一行空行，最后一行源码与/END-FREE 保留一行空行。

第一行源码首字母比/FREE 首字符’/’缩进三个空格，即应该与第一个字母‘E’对齐。必须以‘;’结束，结尾的‘;’必须与之前的非空字符保持一个空格。

同一层次源码的首母必须对齐。

操作符两侧都必须保留一个空格。

源码写在多行，续行的首字母必须缩进三个空格，全部续行的首字母对齐。

注释行以‘//’开始，首字母‘/’应该与被注释行的首字母对齐。注释不能插在语句与续行之间。

条件处理、循环处理内的源码应该缩进三个空格，例如：IF 条件判断，之后语句的首字符必须缩进三个空格，即与 ENDIF 的第四个字母 I 对齐。

条件处理、循环处理可以嵌套，每一层都需要比上一层缩进三个空格。

子例程名必须以‘@SR_’开始。子例程的定义以 BEGSR 开始，以 ENDSR 结束。

子例程前必须有注释，说明子例程的功能。注释的第一行必须与之前的语句保留一行空行。

BEGSR 的首字母‘B’应该比‘/FREE’的首字母缩进三个空格。BEGSR、ENDSR 的首字母必须对齐。

子例程的第一行语句，应该与 BEGSR 间保持一行空行；子例程的最后一行语

句，应该与 ENDSR 间保持一行空行；子例程的第一行语句首字母应该比 BEGSR 缩进三个空格，即与 BEGSR 的‘S’对齐。

子例程中的语句也需要按判断、循环的层次缩进三个空格，与主过程度的方法相同。

子例程中同一层次的首字母必须对齐。

7.2 变量赋值

在自由格式 RPG 中不再使用 MOVE、MOVEL、Z-ADD、Z-SUB 对变量赋值。

而是直接使用 EVAL、EVALR 对变量赋值。

字符串赋值时 EVAL 功能类似于 MOVEL（左对齐），但不保留尾部的数据，会直接赋空；

字符串赋值 EVALR 功能类似于 MOVE（右对齐），但不保留首部的数据，会直接赋空；

字符的拼接可以直接使用+运算。

EVAL、EVALR 不支持数据类型转换。

必须通过%CHAR、%EDITC、%DEC、%DATE、%TIME、%TIMESTAMP 进行数据类型转换。也可以借助数据结构定义，实现数据类型转换。

一般情况下，可以省略 EVAL，直接使用‘=’。

7.3 文件操作

自由格式 RPG 不再支持使用指示器判断文件操作结果，必须使用系统提供的%BIF 进行判断。

为了方便开发使用，给出了操作码与%BIF 的对照表。

功能说明	操作码	操作对象	判断%BIF	错误处理
定位	SETLL	FILE	%EQUAL %FOUND	(E),%ERROR
	SETGT	FILE	%FOUND	(E),%ERROR
读文件	READ	FILE	%EOF	(E),%ERROR
	READE	FILE	%EOF	(E),%ERROR

	READP	FILE	%EOF	(E),%ERROR
	READPE	FILE	%EOF	(E),%ERROR
随机查找	CHAIN	FILE	%FOUND	(E),%ERROR
删除	DELETE	FILE	%FOUND	(E),%ERROR
更新	UPDATE	RECORD		(E),%ERROR
增加	WRITE	RECORD	%EOF	(E),%ERROR
释放锁	UNLOCK	FILE		(E),%ERROR
打开文件	OPEN	FILE		(E),%ERROR
关闭文件	CLOSE	FILE		(E),%ERROR

虽然大部分操作码同时支持文件 FILE 和记录格式 RECORD，但为了系统的统一。UPDATE、WRITE 使用记录格式，其他操作一律使用文件。

以上操作码都支持(E)的处理，应该根据需要在程序使用(E)，并对%ERROR 进行判断，以捕获错误。在多进程并行的情况可以很容易捕获记录锁、键字重复等状态，以保证处理流程的正确。

例如：只希望处理数据库中等于某一键字的数据，可以使用 SETLL 和%EQUAL 判断是否有需要的数据，再使用 READE 读数据。

```
K01_XXXCODE = '0001' ;
SETLL(E) %KDS(DS_TTXXX_K01) TTXXXL01 ;
IF NOT %ERROR AND %EQUAL ;
   READE(E) %KDS(DS_TTXXX_K01) TTXXXL01 ;
   DOW NOT %ERROR AND NOT %EOF ;
      // 处理流程
      READE(E) %KDS(DS_TTXXX_K01) TTXXXL01 ;
   ENDDO ;
ENDIF ;
```

如果希望某一键字范围的数据，可以使用 SETLL 和%FOUND 判断是否有需要的数据，再使用 READ 读数据。

```
K01_XXXCODE = '0001' ;
SETLL(E) %KDS(DS_TTXXX_K01) TTXXXL01 ;
IF NOT %ERROR AND %FOUND ;
   READ(E) TTXXXL01 ;
   DOW NOT %ERROR AND NOT %EOF AND XXXCODE <= '0009' ;
      // 处理流程
      READ(E) TTXXXL01 ;
   ENDDO ;
ENDIF ;
```

实际程序中如果使用(E)捕获异常，则必须根据需要增加相应的处理流程。

7.4 数值计算

在自由格式中可以使用表达式完成复杂的计算。直接使用+、一、*、/进行计算，不再使用操作码。

表达式可以续行，续行的首字符缩进三个空格。如果有多行续行，则续行首字母应该对齐。

7.5 日期计算

自由格式 RPG 不支持使用 TIME 获取日期、时间、日期+时间。自由格式必须使用%BIF：%DATE、%TIME、%TIMESTAMP 获取相应的日期、时间、时间戳，接收的变量必须是相应的数据类型。

日期变量类型转换：

日期转换为字符：

```
C_DATE = %CHAR(%DATE():*ISO) ;//返回为 YYYY-MM-DD
C_DATE = %CHAR(%DATE():*ISO0) ;//返回为 YYYYMMDD
```

字符转换为日期：

```
D_DATE = %DATE( ‘2012-04-01’ :*ISO) ;
D_DATE = %DATE( ‘20120401’ :*ISO) ;
```

日期转换为数值：（V6R1 可以直接转换，V5R4 通过字符）

```
P_DATE = %DEC(%CHAR(%DATE():*ISO0):8:0) ;
```

(建议借用数据结构方式进行转换，提高运行效率)

数值转换为日期：

```
D_DATE = %DATE(20120401) ;
```

取日期值中的年、月、日：

```
YEAR = %SUBDT(%DATE():*Y) ;
```

MONTH = %SUBDT(%DATE():*M) ;

DAY = %SUBDT(%DATE():*D) ;

日期计算：

D_DATE = %DATE() + %YEARS(1) + %MONTHS(2) + %DAYS(3) ;//计算当前日期一年又两月又三天后的日期;

D_DATE = %DATE() - %MONTHS(2) ;//计算当前日期两月前的日期;

时间处理、时间戳处理与日期类似。

7.6 嵌入 SQL

在自由格式 RPG 中可以直接嵌入 SQL，不需要使用固定格式 RPG。

使用方法：

EXEC SQL SQL 命令;

可以使用续行，续行方法与普通 RPG 语句相同。

7.7 交易服务组件的一般处理流程

7.7.1 数据初始化

7.7.1.1 接收组件参数

必须接收 TWA 参数，使用以下语句：

PTR_TWA = %ADDR(PI_TWA) ;

必须接收组件接口参数，使用以下语句：

PTR_zznnnnI = %ADDR(PI_zznnnnI) ;

如果有不循环接口定义，则需要为接口分配地址：

PTR_zznnnnIS = %ADDR(zznnnnI.IS) ;

PTR_zznnnnES = %ADDR(zznnnnI.ES) ;

如果有循环接口定义，则需要为接口分配地址：

PTR_zznnnnIM = %ADDR(zznnnnI.IM) ;

PTR_zznnnnEM = %ADDR(zznnnnI.EM) ;

如果交易服务组件有标准输出（即定义了 zznnnnE），则必须将 TWA 中的组件输出区指针指向组件输出数据结构，使用以下语句：

PTR_zznnnnE = SCCTWA.I_OUTP ;

清空输出接口：

CLEAR zznnnnE ;

7.7.1.2 程序变量初始化

根据程序需要编写

7.7.1.3 文件指针初始化

根据程序需要编写

7.7.2 数据检查

可以在服务组件程序中直接检查数据。

也可以调用其他组件进行数据检查，调用组件时使用以下方法：

CALLP zzxxxxxxR(SCCTWA : zzxxxxxx)

或

CALLP(E) zzxxxxxxR(SCCTWA : zzxxxxxx)

其中 ：

zzxxxxxxR 为组件程序名称；

zzxxxxxx 为组件接口名称，与组件程序名称的前 8 位相同。

全部数据检查后，应该判断 TWA 的 MSGT 是否为‘E’，如果为‘E’表示检查发生错误，必须结束程序。

7.7.3 数据处理

可以在服务组件程序中直接进行数据处理。

也可以其他组件进行数据处理，调用组件时使用以下方法：

CALLP zzxxxxxxR(SCCTWA : zzxxxxxx)

或

CALLP(E)　zzxxxxxxR(SCCTWA : zzxxxxxx)

其中 ：

zzxxxxxxR 为组件程序名称；

zzxxxxxx 为组件接口名称，与组件程序名称的前 8 位相同。

7.7.4 输出处理

如果组件只有标准输出，即只通过 zznnnnE 格式输出，则不需要进行特殊处理。平台会自动将输出的内容返回，输出的指针已经在初始化中获得。

如果是多页查询，则需要使用技术平台的功能 F_SCOUTPT 进行输出处理，‘多页标志’应该设置为‘Y’。

7.7.5 调用组件的判断

调用单元组件后，根据 TWA 的 MSGT 判断是否正确，为‘E’表示单元组件抛了错误信息。

调用函数组件后，根据返回 RTCD 判断是否正确，如果 RTCD 的第 3 位为‘R’表示错误信息，组件可以抛出函数组件返回的信息，应该同时抛出 RTCD、AINF。

7.7.6 错误返回方式

服务组件处理过程中如果发生错误，调用 F_SCSDMSG 发送自身的错误信息，信息代码必须使用变量 XC_MGID、附加信息必须使用变量 XC_AINF，将组件名称赋给附加信息字段，并且必须占用附加信息的前 10 位。

7.8 其他服务组件的一般处理流程

7.8.1 数据初始化

7.8.1.1 接收组件参数

必须接收 TWA 参数，使用以下语句：

PTR_TWA = %ADDR(PI_TWA) ;

必须接收组件接口参数，使用以下语句：

PTR_zzxxxxxx = %ADDR(PI_zzxxxxxx) ;

为不循环接口分配地址：

PTR_zzxxxxxxIS = %ADDR(zzxxxxxx.IS) ;

PTR_zzxxxxxxES = %ADDR(zzxxxxxx.ES) ;

清空输出接口：

CLEAR zzxxxxxx.ES ;

为循环接口分配地址：

PTR_zzxxxxxxIM = %ADDR(zzxxxxxx.IM) ;

PTR_zzxxxxxxEM = %ADDR(zzxxxxxx.EM) ;

清空输出接口：

CLEAR zzxxxxxx.EM ;

7.8.1.2 程序变量初始化

根据程序需要编写

7.8.1.3 文件指针初始化

根据程序需要编写

7.8.2数据检查

可以在服务组件程序中直接检查数据。

也可以其他组件进行数据检查，调用组件时使用以下方法：

CALLP zzxxxxxxR(SCCTWA : zzxxxxxx)

或

CALLP(E) zzxxxxxxR(SCCTWA : zzxxxxxx)

其中 ：

zzxxxxxxR 为组件程序名称；

zzxxxxxx 为组件接口名称，与组件程序名称的前 8 位相同。

全部数据检查后，应该判断 TWA 的 MSGT 是否为‘E’，如果为‘E’表示检查发生错误，必须结束程序。

7.8.3数据处理

可以在服务组件程序中直接进行数据处理。

也可以调用其他组件进行数据处理，调用组件时使用以下方法：

CALLP zzxxxxxxR(SCCTWA : zzxxxxxx)

或

CALLP(E) zzxxxxxxR(SCCTWA : zzxxxxxx)

其中 ：

zzxxxxxxR 为组件程序名称；

zzxxxxxx 为组件接口名称，与组件程序名称的前 8 位相同。

7.8.4调用组件的判断

调用单元组件后,根据 TWA 的 MSGT 判断是否正确，为‘E’表示单元组件抛了错误信息。

调用函数组件后，根据返回 RTCD 判断是否正确，如果 RTCD 的第 3 位为‘R’表示错误信息，组件可以抛出函数组件返回的信息，应该同时抛出 RTCD、AINF。

7.8.5错误返回方式

服务组件处理过程中如果发生错误，调用 F_SCSDMSG 发送自身的错误信息，信息代码必须使用变量 XC_MGID、附加信息必须使用变量 XC_AINF，将组件名称赋给附加信息字段，并且必须占用附加信息的前 10 位。

7.9 单元组件、函数组件处理流程

7.9.1数据初始化

7.9.1.1 接收组件参数

必须接收 TWA 参数，使用以下语句：

PTR_TWA = %ADDR(PI_TWA) ;

必须接收组件接口参数，使用以下语句：

PTR_zzxxxxxx = %ADDR(PI_zzxxxxxx) ;

为不循环接口分配地址：

PTR_zzxxxxxxIS = %ADDR(zzxxxxxx.IS) ;

PTR_zzxxxxxxES = %ADDR(zzxxxxxx.ES) ;

清空输出接口：

CLEAR zzxxxxxx.ES ;

为循环接口分配地址：

PTR_zzxxxxxxIM = %ADDR(zzxxxxxx.IM) ;

PTR_zzxxxxxxEM = %ADDR(zzxxxxxx.EM) ;

清空输出接口：

CLEAR zzxxxxxx.EM ;

7.9.1.2 程序变量初始化

根据程序需要编写

7.9.1.3 文件指针初始化

根据程序需要编写

7.9.2数据检查

可以在组件程序中直接检查数据。

也可以调用单元组件、函数组件进行数据检查，调用组件时使用以下方法：

CALLP zzxxxxxxR(SCCTWA : zzxxxxxx)

或

CALLP(E) zzxxxxxxR(SCCTWA : zzxxxxxx)

其中 ：

zzxxxxxxR 为组件程序名称；

zzxxxxxx 为组件接口名称，与组件程序名称的前 8 位相同。

函数组件不允许调用单元组件。

全部数据检查后，应该判断 TWA 的 MSGT 是否为‘E’，如果为‘E’表示检查发生错误，必须结束程序。

7.9.3数据处理

可以在组件程序中直接进行数据处理。

也可以调用单元组件、函数组件进行数据处理，调用组件时使用以下方法：

CALLP　zzxxxxxxR(SCCTWA : zzxxxxxx)

或

CALLP(E)　zzxxxxxxR(SCCTWA : zzxxxxxx)

其中 ：

zzxxxxxxR 为组件程序名称；

zzxxxxxx 为组件接口名称，与组件程序名称的前 8 位相同。

函数组件不允许调用单元组件。

7.9.4调用组件的判断

调用单元组件后,根据 TWA 的 MSGT 判断是否正确，为‘E’表示单元组件抛了错误信息。

调用函数组件后，根据返回 RTCD 判断是否正确，如果 RTCD 的第 3 位为‘R’表示错误信息，单元组件可以抛出函数组件返回的信息，应该同时抛出 RTCD、AINF。

7.9.5错误返回方式

单元组件处理过程中如果发生错误，调用 F_SCSDMSG 发送自身的错误信息，信息代码必须使用变量 XC_MGID、附加信息必须使用变量 XC_AINF，将组件名称赋给附加信息字段，并且必须占用附加信息的前 10 位。

函数组件只能通过返回码返回错误，必须同时返回附加信息，将组件名称赋给附加信息字段，并且必须占用附加信息的前 10 位。返回码、附加信息定义在组件标准输出接口中，一般情况下字段为：zzxxxxxxES.ESRTCD 、zzxxxxxxES.ESAINF，其中 zzxxxxxx 为组件接口名称。

7.10 错误处理

7.10.1 使用(E)方式捕获异常

RPG 的大部分操作码都支持(E)方式。应该根据需要确定是否使用(E)方式，就必须对%ERROR 进行判断，并且编写发生错误时的处理代码。

如果程序无法确定发生错误如何处理，就不要使用(E)方式，由平台或系统捕获错误。否则，相当于隐藏了错误，可能产生不可预见的错误。

7.10.2 使用MONITOR方式捕获异常

RPG 的部分操作码不支持(E)方式。例如：EVAL、EVALR。

如果发生除 0 错误、数字类型错误等就不能使用(E)方式捕获异常。

自由格式提供了 MONITOR/ON-ERROR/ENDMON 这组操作码，可以捕获 MONITOR/ENDMON 之间语句的错误，由 ON-ERROR 判断%STATUS 的值并进行相应处理。也可以在 ON-ERROR 不判断%STATUS 的值，相当于捕获全部异常。

```
MONITOR ;
   A = B / C ;
   ON-ERROR ;
      //错误处理流程;
ENDMON ;
```

7.10.3 使用*PSSR方式捕获异常

系统提供了*PSSR 的方式，只要在 RPG 中定义了*PSSR 子例程序，程序发生了没有监控的异常，则系统会自动调用*PSSR 子例程序。

在*PSSR 子例程序中可以记录异常的原因、状态等信息，并进行处理。

一般程序使用公用定义的*PSSR。在程序结束处使用/COPY 复制引用。

标准的*PSSR 定义源文件：C@@DFNSRC，源成员：SCPUBSUB1R。

源码内容：

```
/IF NOT DEFINED(@@PUBSUB1R)
/DEFINE @@PUBSUB1R


**********************************************************************
```

```
     * 例程名称：*PSSR
     * 功能描述：异常处理子过程
     *****************************************************************
     /FREE
        BEGSR *PSSR ;
           SCCEXCPT.EXCPTTYPE = G_EXCPT_TYPE_D ;
           SCCEXCPT.EXCPTPROG = XC_SYPN ;
           SCCEXCPT.EXCPTSRCL = %DEC(XC_LINE:8:0) ;
           SCCEXCPT.EXCPTFUNC = DS_FILE.XC_OPCD ;
           SCCEXCPT.EXCPTOBNM = XC_FILE ;
           XPGM_SCEXCPT(SCCTWA:SCCEXCPT) ;
        ENDSR ;
     /END-FREE

     /ENDIF
```

程序的使用方法：

必须以下复制公用定义：

系统程序反馈信息结构、文件反馈信息结构定义：@@DTADFN1H;

例外处理组件接口数据结构定义：SCOLPDFN2H

例外处理组件调用原型定义：SCOLPDFN3H

*PSSR 子例程序的公用定义：SCPUBSUB1R

```
FTTXXXL00  IF   E           K DISK    INFDS(DS_FILE)
F                                     INFSR(*PSSR)

 *****************************************************************
 *组件原型定义
 *****************************************************************
```

```
D PGM_TTXXXXXX     PR                  EXTPGM('TTXXXXXXR')
D  PR_TWA                              LIKEDS(XDS_SCCTWA)
D  PR_TTCXXXXXXX                       LIKEDS(TTCXXXXXX)
D PGM_TTXXXXXX     PI
D  PI_TWA                              LIKE(XDS_SCCTWA)
D  PI_TTXXXXXX                         LIKE(TTXXXXXX)

 *****************************************************************
 *数据结构定义
 *****************************************************************
 * TWA 定义
D SCCTWA           DS                  LIKEDS(XDS_SCCTWA)
D                                      BASED(PTR_TWA)
 *异常处理结构定义
D SCCEXCPT         DS                  LIKEDS(XDS_SCCEXCPT)

D DS_FILE          DS                  LIKEDS(XDS_FILE)
 *组件接口定义（应该参考外部定义建立）
D TTCXXXXXX        DS                  EXNAME(TTCXXXXXX)
D                                      QUALIFIED
D                                      BASED(PTR_TTXXXXXX)

 *****************************************************************
 *复制公用定义
 *****************************************************************
 /COPY CDFNSRC,@@DTADFN1H
 /COPY CDFNSRC,SCDTADFN3H
 /COPY CDFNSRC,SCPUBDFN3H
 /FREE
```

```
        //数据初始化子过程
        EXSR @SR_INZDTA ;

        //程序处理逻辑

        //结束处理子过程
        EXSR @SR_RETURN ;

        // ------------------------------------------------------
        // 数据初始化子过程
        // ------------------------------------------------------
        BEGSR @SR_INZDTA ;

           //获取 TWA 指针
           PTR_TWA = %ADDR(PI_TWA) ;
           PTR_TTXXXXXX = %ADDR(PI_TTXXXXXX) ;

        ENDSR ;

        // ------------------------------------------------------
        // 结束处理子过程
        // ------------------------------------------------------
        BEGSR @SR_RETURN ;

           RETURN ;

        ENDSR ;

     /END-FREE
```

```
*复制公用*PSSR

/COPY CDFNSRC,@@PUBSUB1R
```

8 过程、函数

8.1 过程、函数的使用原则

1) 函数、过程名应准确描述函数、过程的功能，使用动宾词组为执行某操作的函数、过程命名。
2) 明确函数或过程的功能，一个函数或过程仅完成一件功能，避免函数或过程功能过于复杂。防止把没有关联的功能放到一个函数或过程中。
3) 明确规定对函数或过程接口参数的合法性检查应由调用者负责还是由函数或过程本身负责，缺省是由调用者负责。
4) 防止将函数或过程的接口参数作为工作变量，数据内容被误修改。不应该被函数或过程修改的接口参数，应该定义为 CONST 或 VALUE 方式接收。
5) 函数、过程的返回值要清楚、明了。
6) 在调用函数、过程填写参数时，应尽量减少没有必要的默认数据类型转换或强制数据类型转换。
7) 对所调用函数或过程的错误返回码要仔细、全面地处理。

8.2 过程、函数定义方法

8.2.1 开始、结束定义

过程名称：F_zzxxxxxx，其中：

F_：表示过程或公用函数

zz：应用模块标识；

xxxxxx：过程名称；

过程开始定义为：P 过程名称……B，如果是服务程序的函数，还需要定义关键字 EXPORT；

过程结束定义为：P 过程名称……E；

在过程开始定义前，需要增加过程注释，内容包括：过程名称、功能描述、入口参数、返回参数。

定义方法：

```
     ****************************************************************
     * 函数名称：F_SCSDMSG
     * 功能描述：抛信息处理
     * 接口参数：
     *      01    *           TWA 接口
     *      02   7A           信息代码
     *      03   5P0          字段序号
     *      04  60A           附加信息
     *      05   1A           继续标志
     *      06   1A           主控标志
     * 返回参数：
     *      01   1A            结束标志
     ****************************************************************
    P F_SCSDMSG        B                   EXPORT
     *过程内容
    P F_SCSDMSG        E
```

8.2.2过程接口定义

过程接口在 D 表中定义，类似于程序的接口定义，以 PI 的方式定义，如果有返回值则定义数据长度及类型，如果没有返回值不定义。在 PI 之后定义过程的入口参数，参数个数、类型应该与过程原型定义匹配。程序可以直接使用这些参数。

定义方法：

```
     D F_SCCLCPN        PI             1
     D  PTR_TWA                         *
     D  C_PGMN                        10A   CONST
     D  PTR_COM                         *   CONST
     D  P_COML                        5P 0 CONST
     D  PTR_MSG                         *   OPTIONS(*OMIT)
     D  P_MSGN                        5P 0 CONST
     D                                     OPTIONS(*OMIT)
     D  P_FLDN                        5P 0 CONST
```

```
D                                       OPTIONS(*OMIT)
D  C_CNTF                          1A   CONST
D                                       OPTIONS(*OMIT)
D  C_OUTF                          1A   CONST
D                                       OPTIONS(*OMIT)
D  C_HDLF                          1A   CONST
D                                       OPTIONS(*OMIT)
```

8.2.3过程变量定义

过程的变量定义与程序的变量定义类似。

定义方法：

```
D SCCTWA          DS                    LIKEDS(XDS_SCCTWA)
D                                       BASED(PTR_SCCTWA)
D SCCCLCPN        DS                    LIKEDS(XDS_SCCCLCPN)
```

8.2.4处理过程定义

处理过程定义与程序定义类似，不使用*INLR＝*ON 结束过程，直接使用RETURN 返回。

如果接口定义（PI）有返回值，则 RETURN +返回值结束过程。返回值可以是变量也可以是常量，但数据类型必须与 PI 定义的类型一致。

定义方法：

```
 /FREE
    PTR_SCCTWA = PTR_TWA ;
    SCCCLCPN.CLCPNPROG = C_PGMN ;
    SCCCLCPN.CLCPNCOMP = PTR_COM ;
    SCCCLCPN.CLCPNCOML = P_COML ;
    IF %ADDR(PTR_MSG) <> *NULL ;
       SCCCLCPN.CLCPNMSGP = PTR_MSG ;
    ELSE ;
       SCCCLCPN.CLCPNMSGP = *NULL ;
    ENDIF ;
    IF %ADDR(P_MSGN) <> *NULL ;
       SCCCLCPN.CLCPNMSGN = P_MSGN ;
    ELSE ;
       SCCCLCPN.CLCPNMSGN = *ZERO ;
    ENDIF ;
```

```
        IF %ADDR(P_FLDN) <> *NULL ;
           SCCCLCPN.CLCPNFLDN = P_FLDN ;
        ELSE ;
           SCCCLCPN.CLCPNFLDN = *ZERO ;
        ENDIF ;
        IF %ADDR(C_CNTF) <> *NULL ;
           SCCCLCPN.CLCPNCNTF = C_CNTF ;
        ELSE ;
           SCCCLCPN.CLCPNCNTF = *BLANK ;
        ENDIF ;
        IF %ADDR(C_OUTF) <> *NULL ;
           SCCCLCPN.CLCPNOUTF = C_OUTF ;
        ELSE ;
           SCCCLCPN.CLCPNOUTF = *BLANK ;
        ENDIF ;
        IF %ADDR(C_HDLF) <> *NULL ;
           SCCCLCPN.CLCPNHDLF = C_HDLF ;
        ELSE ;
           SCCCLCPN.CLCPNHDLF = *BLANK ;
        ENDIF ;
        XPGM_SCCLCPN(SCCTWA:SCCCLCPN) ;
        RETURN SCCTWA.TWAEXCF ;
      /END-FREE
```

9 服务程序的编程规范

9.1 服务程序模块的定义

服务程序模块的定义方法与程序相同。

只是编译时不生成*PGM，而是生成*MODULE。

9.2 服务程序函数原型定义

一个服务程序的全部函数原型应该定义在同一个源成员中，源成员名称为：服务程序名称+H；保存系统公用源文件中，源文件名称为：C@@DFNSRC。

为了防止程序重复引用，应用使用 DEFINED 方式定义：

```
/IF NOT DEFINED(源成员名称)
/DEFINE 源成员名称
.....
/ENDIF
```

原型定义包括两部分：

函数说明：函数名称、函数功能、入口参数、返回参数

函数原型：函数名称及返回，入口参数个数及类型

定义方法：

```
*****************************************************************
* 函数名称：F_SCSDMSG
* 功能描述：抛信息处理
* 入口参数：
*      01    *           TWA 接口
*      02   7A           信息代码
*      03   5P0          字段序号
*      04  60A           附加信息
*      05   1A           继续标志
*      06   1A           主控标志
* 返回参数：
*      01   1A            结束标志
*****************************************************************
D F_SCSDMSG       PR              1
```

```
D  FR_TWAP                    *
D  FR_MSCD                    7A    CONST
D  FR_FLDN                    5P 0  CONST
D                                   OPTIONS(*OMIT)
D  FR_AINF                    60A   CONST
D                                   OPTIONS(*OMIT)
D  FR_CNTF                    1A    CONST
D                                   OPTIONS(*OMIT)
D  FR_CNSF                    1A    CONST
D                                   OPTIONS(*OMIT)
```

9.3 服务程序定义

一个服务程序建立一个源成员，源成员的名称就是服务程序的名称，服务程序的名称：zzPUBSRVn。源成员应该保存在服务程序专用的源文件中，源文件名称：CzzSRVSRC。

服务程序由多个函数定义组成，应该包含服务程序的全部函数定义。每个函数定义就是一个过程定义，与过程定义方法相同，只需要使用关键字 EXPORT。

9.4 服务程序联编定义

服务程序的联编定义是声明服务程序 EXPORT 的函数。服务程序联编定义是定义在一个独立的源成员中，源成员名称：服务程序名称+‘B’，源成员类型：BND，源成员保存在系统公用的源文件中，源文件名称：C@@BNDSRC。

定义方法：

```
STRPGMEXP   SIGNATURE('SCPUBSRV1')

EXPORT      SYMBOL('F_SCSDMSG')
EXPORT      SYMBOL('F_SCCLCPN')
EXPORT      SYMBOL('F_SCRPARM')
EXPORT      SYMBOL('F_SCOUTPT')
EXPORT      SYMBOL('F_SCTRACE')

ENDPGMEXP
```

在 STRPGMEXP 定义行，为保证在以后维护服务程序时不重新编译全部应用程

序，必须定义 SIGNATURE（），内容为服务程序名称。

EXPORT 定义服务程序对外提供的函数清单，函数名中字母大小写必须过程定义的一致。

在维护服务程序时，只能将新的 EXPORT 函数增加在现有内容的后面，不允许删除已有的 EXPORT 内容，也不能调整已有的 EXPORT 内容的顺序。

9.5 服务程序建立方法

服务程序建立过程比较复杂：

将服务程序所有功能建立为*MODULE；

将服务程序建立为*MODULE；

将这些模块联编成*SRVPGM，建立是使用联编定义。

使用 BNDDIR 辅助完成联编，服务程序的联编目录名称：CNCBMODBND，联编目录在 LIB：CNCBMODD 中。

应用系统提供工具进行统一编译，完成以上功能。提高开发效率，避免发生错误。

9.6 程序使用服务程序

需要在程序源码中使用/COPY 复制服务程序函数的原型定义；

在程序中直接使用服务程序提供的函数：F_zzxxxxxx；

编译生成时与服务程序联编，程序的联编目录名称：CNCBPGMBND，联编目录在 LIB：CNCBMODD 中。

应用系统提供工具进行统一编译，完成以上功能。提高开发效率，避免发生错误。

程序样例：

```
FTTXXXL01  IF   E           K DISK    INFDS(DS_FILE)
F                                     INFSR(*PSSR)
```

```
     *****************************************************************
     *组件调用接口定义
     *****************************************************************
D PGM_TTXXXXXX     PR                  EXTPGM('TTXXXXXXR')
D   PR_TWA                             LIKE(SCCTWA)
D   PR_TTXXXXXX                        LIKE(TTXXXXX)
D PGM_TTXXXXXX     PI
D   C_TWA                              LIKE(SCCTWA)
D   C_TTXXXXXX                         LIKE(TTXXXXXX)


     *****************************************************************
     *数据结构及变量定义
     *****************************************************************
     * TWA 定义
D SCCTWA           DS                  LIKEDS(XDS_SCCTWA)
D                                      BASED(PTR_TWA)

     *异常处理结构定义
D SCCEXCPT         DS                  LIKEDS(XDS_SCCEXCPT)

D DS_FILE          DS                  LIKEDS(XDS_FILE)
     *组件接口定义（应该参考外部定义建立）
D TTXXXXXX       E DS                  EXNAME(TTXXXXXX)
D                                      QUALIFIED
D                                      BASED(PTR_TTXXXXXX)
D DS_XXX_K01     E DS                  EXNAME(TTXXXL00:*KEY)
D                                      PREFIX(K01_)

     /COPY C@@DFNSRC,@@DTADFN1H
```

```
/COPY C@@DFNSRC,SCDTADFN1H
/COPY C@@DFNSRC,SCDTADFN2H
/COPY C@@DFNSRC,SCDTADFN3H
/COPY C@@DFNSRC,SCPUBDFN3H
/COPY C@@DFNSRC,SCPUBSRV1H
/FREE
   //数据初始化子过程
   EXSR @SR_INZDTA ;

   //检查键字
   CHAIN(E) %KDS(DS_TTXXX_K01) TTXXXL01 ;
   IF NOT %ERROR AND %FOUND ;
      SCCTESTB.TESTBDESC = '文件内容' ;
   ELSE ;
      IF F_SCSDMSG(PTR_TWA:'SCE9903':*OMIT:*OMIT:G_SDMSG_CNTF_Y:
         *OMIT) = G_TWA_EXCF_Y ;
         EXSR @SR_RETURN ;
      ENDIF ;
   ENDIF ;

   //检查其他信息
   //检查处理流程
   IF NOT %ERROR ;
      IF F_SCSDMSG(PTR_TWA:'SCE9904':*OMIT:*OMIT:G_SDMSG_CNTF_Y:
         *OMIT) = G_TWA_EXCF_Y ;
         EXSR @SR_RETURN ;
      ENDIF ;
   ENDIF ;
```

```
        //结束处理子过程
        EXSR @SR_RETURN ;

        // ------------------------------------------------------------
        // 数据初始化子过程
        // ------------------------------------------------------------
        BEGSR @SR_INZDTA ;
           //获取 TWA 指针
           PTR_TWA = %ADDR(C_TWA) ;
           PTR_TTXXXXXX = %ADDR(C_TTXXXXXX) ;
        ENDSR ;

        // ------------------------------------------------------------
        // 结束处理子过程
        // ------------------------------------------------------------
        BEGSR @SR_RETURN ;
           RETURN ;
           *INLR = *ON ;
        ENDSR ;

     /END-FREE
     /COPY C@@DFNSRC,@@PUBSUB1H
```

10 程序的质量保证

1) 在软件设计过程中构筑软件质量。
2) 代码质量保证优先原则
   a) 正确性，指程序要实现设计要求的功能。
   b) 稳定性、安全性，指程序稳定、可靠、安全。
   c) 可测试性，指程序要具有良好的可测试性。
   d) 规范/可读性，指程序书写风格、命名规则等要符合规范。
   e) 全局效率，指软件系统的整体效率。
   f) 局部效率，指某个模块/子模块/函数的本身效率。
3) 认真处理程序所能遇到的各种出错情况。
4) 系统运行之初，要初始化有关变量及运行环境，防止未经初始化的变量被引用。
5) 严禁随意更改其它模块或系统的有关设置和配置。
6) 不能随意改变与其它模块的接口。
7) 充分了解系统 API 的功能特性后，合理运用系统 API 实现程序处理功能。
8) 编程时，要注意边界条件，特别是防止差 1 错误。
9) 结合实际需求，IF 语句尽量加上 ELSE 分支，SELECT 语句尽量加上 OTHER 分支，对没有 ELSE 或 OTHER 分支的语句要小心对待。
10) 精心构造算法，并对其性能、效率进行测试。
11) 对较关键的算法最好使用其它算法来验算。
12) 时刻注意表达式是否会溢出。
13) 使用变量时要注意边界值。
14) 注意数据、内容等是否越界。字符串操作是否越界。
15) 系统应具有一定的容错能力，对一些错误事件能进行自动补救。
16) 对一些操作代码要增加容错处理，防止对影响数据安全、准确。以提高系统的安全性。

11 程序的测试

11.1 可测性要求

1) 同一项目组或平台应该有统一的为集成测试及系统联调准备的测试工具，并且要有详细的说明。
2) 在编写代码之前，应预先设计好程序调试与测试的方法和手段，并设计好各种调测开关及相应测试代码如写调测日志、输出 DUMP 信息等。
3) 编程的同时要为单元测试选择恰当的测试点，并仔细构造测试代码、测试用例，同时给出明确的注释说明。测试代码部分应作为（模块中的）一个子模块，以方便测试代码在模块中的安装与拆卸(通过调测开关)。
4) 编写防错程序（即错误出口程序），然后在处理错误之后可用断言宣布发生错误。
5) 在进行集成测试/系统联调之前，要构造好测试环境、测试项目及测试用例，同时仔细分析并优化测试用例，以提高测试效率。
6) 用调测开关来切换软件的 DEBUG 版和正式版，而不要同时存在正式版本和 DEBUG 版本的不同源文件，以减少维护的难度。
7) 软件的 DEBUG 版本和发行版本应该统一维护，不允许分家，并且要时刻注意保证两个版本在实现功能上的一致性。
8) 设置不同级别和类型的调测开关。在软件系统中设置与取消有关测试手段，不能对软件实现的功能等产生影响。
9) 正式软件产品中应把断言及其它调测代码去掉（即把有关的调测开关关掉）。

11.2 程序测试原则

1) 代码必须经过单元测试，单元测试要覆盖语句。
2) 仔细设计并分析测试用例，使测试用例覆盖尽可能多的情况，以提高测试用例效率。

3）测试用例必须考虑数据、变量的边界值，并包含这些情况。

4）关键代码如果测试用命无法覆盖，则必须通过 DEBUG 跟踪测试。

5）保留测试信息，包括：测试用例数据、测试前数据、测试后数据，以便分析、总结，并可以反复测试。

6）测试中出现的问题，必须寻找问题的根本原因，并进行修改，并以相同的数据进行验证，以确保问题不再发生。

7）代码维护、优化、升级要有详细文档，要经过审查及测试。

8）代码维护、优化、升级后，必须使用以往的测试数据进行验证，并应该根据新的功能调整测试用，保证测试用例的正确。

11.3 测试程序使用

每个模块都在源码库 CNCBSRCD 中建立专用的测试源文件：CzzTSTSRC。

为组件开发测试程序，测试程序名称为：组件程序名称+‘Z’。

测试程序目标保存在 CNCBTSTD 中，可以用于反复测试需要。

12 程序效率

12.1 编程应该遵守的原则

1) 编程时要注意代码的效率。在保证软件系统的正确性、稳定性、可读性及可测性的前提下，提高代码效率。
2) 通过对系统数据结构的划分与组织的改进，以及对程序算法的优化来提高空间效率。
3) 仔细分析有关算法，并进行优化。
4) 仔细考查、分析系统及模块处理输入（如事务、消息等）的方式，并加以改进。
5) 对模块中函数及过程的划分及组织方式进行分析、优化，改进模块中函数及过程的组织结构，提高程序效率。
6) 在保证程序质量的前提下，通过压缩代码量、去掉不必要代码以及减少不必要的局部和全局变量，来提高空间效率。
7) 仔细构建服务程序（Service Program），编写调用频繁或性能要求极高的函数。
8) 采用驻留内存的处理方式有助于提高程序处理速度，但需要注意变量初始、文件指针初始化、资源互锁等问题。
9) 循环体内工作量最小化。在多重循环中，应将最忙的循环放在最内层。尽量减少循环嵌套层次。
10) 尽量避免循环体内含判断语句，应将循环语句置于判断语句的代码块之中。

12.2 提高程序效率的办法

12.2.1 使用RETURN方式结束程序

RPG 程序使用 RETURN 结束程序，代替*INLR = *ON 方式结束程序。

这种方式可以有两个功能：

1）程序驻留内存，提高程序的处理速度；

2）减少文件打开、关闭，达到减少 I/O 次数的目的，提高程序性能。

使用这种方式，必须注意三点问题：

1） 变量初始化：再次调用程序时，系统不再对程序变量做初始化处理，变量保持上次运行时的值。程序必须显性地对变量进行初始化处理（注意：数据结构也是变量的一种，不能忽略），方法是：字符型变量赋空、数字型变量赋零、使用 CLEAR 或 RESET 初始化数据结构。

2） 文件指针：再次调用程序时，由于不再重新打开文件，文件指针仍然停留在处理的位置。需要使用 SETLL、SETGT、CHAIN 等命令重新定位文件指针。

3） 资源锁：程序在处理过程中以锁方式获取数据，但没有使用 UPDATE、DELETE 命令，就直接退出程序，则记录锁不会释放。在程序并发处理中，很容易引起记录锁冲突。需要在程序退出前，即 REUTRN 前，使用 UNLOCK 释放记录锁，可以对全部使用 U 方式打开的文件进行 UNLOCK 处理，UNLOCK 不会产生异常处理。

12.2.2 将动态调用改为静态调用

将使用频繁的公用程序，改为公用函数，再组装成服务程序。

程序与服务程序绑定，调用方式改为静态调用。

12.2.3 使用顺序读取或块读取

如果应用程序不需要按键字处理，可以改为按物理顺序处理，即使用物理文件，这样可以提高文件的访问性能。

如果一定需要按键字处理，可以指定关键字为 BLOCK(*YES)。这种方式不支持 UPDATE、READE、READP、PREADPE，可以采用读数据使用一个文件，更新使用另一个文件，提高读的性能。

12.2.4 分析应用处理瓶颈进行优化

使用厂商提供的系统分析工具，采集系统运行信息，分析应用处理的瓶颈，对应用程序进行优化。

尽量减少循环层次，循环次数。尽量顺序处理，减少随机定位的处理。

13 开发环境说明

13.1 系统标识

系统标识：CNCB，表示中信银行。

13.2 LIB

目标 LIB 名称：CNCBxxxe，其中：xxx 为 LIB 的用途；e 为环境标识，开发环境库 D。

源码 LIB 名称：CNCBSRCzze，其中：zz 为应用模块标识；e 为环境标识，开发环境为 D。

开发环境的 LIB

LIB 名称	LIB 使用说明	备注
CNCBSRCSCD	技术平台源码库	
CNCBSRCBPD	业务平台源码库	
CNCBSRCDPD	负债模块源码库	
CNCBSRCAID	会计模块源码库	
CNCBSRCLND	资产模块源码库	
CNCBSRCTRD	过渡模块源码库	
CNCBSRCCTD	受控源码库	
CNCBOBJD	核心系统目标库	程序目标、显示文件、打印文件
CNCBDTAD	核心系统数据库	生产数据
CNCBCMND	核心系统公用信息库	配置信息
CNCBHSYD	核心系统历史数据库	
CNCBIOFD	核心系统 IOF 库	输入、输出文件结构
CNCBTMPD	核心系统临时数据库	保存查询等处理产生的临时数据文件
CNCBJRND	核心系统日志库	
CNCBSYSD	核心系统系统目标库	在 SBSD、JOBD、JOBQ 等系统目标，这类目标使用程序建立
CNCBMODD	核心系统编译中间库	编译产生的*MODULE 保存这个 LIB 中，BNDDIR 也可以保存这个 LIB 中。如果生产环境直接上目标，则不需要此 LIB
CNCBTSTD	核心系统测试库	存放测试用程序目标，生产环境没

		有此 LIB

13.3 源文件

源文件命名规则：Czzvvvvvv

其中：

C：表示源码

zz：应用模块或子模块标识，为@@时表示应用系统公用源文件

vvvvvv 为源文件用途

开发环境源码库：CNCBSRCzzD

开发环境源文件：

源文件名称	源文件说明	目标对应 LIB	备注
C@@DFNSRC	公用定义源文件	不编译	
C@@DDSSRC	文件定义源文件 保存运行过程中动态建立目标的源码	不编译	
C@@BNDSRC	服务程序联编定义（BND）	不编译	
CzzDDSSRC	文件定义源文件（PF、LF）	CNCBDTAD	
CzzCMNSRC	公用参数源文件（PF、LF）	CNCBCMND	
CzzHSYSRC	历史数据源文件（PF、LF）	CNCBHSYD	
CzzIOFSRC	输入输出源文件（PF、LF）	CNCBIOFD	
CzzTMPSRC	临时文件源文件（PF、LF）	CNCBTMPD	
CzzPGMSRC	程序源文件（DSPF、PRTF、CLLE、RPGLE、C、SQLRPGLE）	CNCBOBJD	编译中间过程产生的*MODULE 放入 xxxMODD SQL 的事务处理为*CHG
CzzMODSRC	程序模块源文件（CLLE、RPGLE、C、SQLRPGLE）	CNCBMODD	只生成*MODULE 的源码 SQL 的事务处理为*CHG
CzzSQLSRC	程序源文件（SQLRPGLE）	CNCBOBJD	编译中间过程产生的*MODULE 放入 xxxMODD SQL 的事务处理为*NONE
CzzSQLMOD	程序模块源文件（SQLRPGLE）	CNCBMODD	只生成*MODULE 的源码 SQL 的事务处理为*NONE
CzzSRVSRC	服务程序源文件（CLLE、RPGLE）	CNCBOBJD	编译中间过程产生的*MODULE 放入 xxxMODD
CzzSYSSRC	系统目标建立程序源文件	CNCBSYSD	系统建立一律使用程序，

	（CLLE、RPGLE）		不允许手工建立。 LIB 使用参数定义，各环境建立程序统一
CzzTSTSRC	测试程序源文件（CLLE、RPGLE）	CNCBTSTD	
CzzBAKSRC	备份源文件	不编译	修改前将源码备份到此源文件，不允许直在源文件中备份

技术平台目前使用的模块代码为 SC，所以技术平台的源文件是以 SC 替换上表中的 zz。

13.4 MSGF

MSGF 名称：CNCBMSGF，所在 LIB：CNCBCMND。

MSGID 的使用规则：zztnnnn

其中：

zz：应用模块标识；

t：信息类别；A：表示授权；P：表示提示；R：表示错误；W：表示警告；

nnnn：信息序号，每一位都可以是 0－9、A－F，具体使用由各应用模块划分。

14 技术平台的使用说明

14.1 技术平台对外提供的功能

技术平台对外提供的内容包含两部分：

公用数据定义，包括：技术平台数据区结构定义、技术平台公用程序接口定义；

公用程序，包括：抛信息、组件调用、取参数、格式输出、跟踪调试、组件调用前处理、组件调用后处理、参数维护、例外处理；

14.2 技术平台公用数据定义

技术平台的公用数据是定义在源成员中。

技术平台为全系统使用公用定义的源码：CNCBSRCSCD，源文件：C@@DFNSRC。

联机平台公用数据定义，源成员：SCOLPDNF1H，定义内容为：

数据名称	说明	公用定义源成员名称	公用定义数据结构名称	程序用数据结构名称
TIA	交易输入区	SCOLPDFN1H	XDS_SCCTIA	SCCTIA
TOA	交易输出区	SCOLPDFN1H	XDS_SCCTOA	SCCTOA
TWA	交易数据区	SCOLPDFN1H	XDS_SCCTWA	SCCTWA
SCA	技术平台专用区	SCOLPDFN1H	XDS_SCCSCA	SCCSCA
CHN	渠道信息数据区	SCOLPDFN1H	XDS_SCCCHN	SCCCHN
SCCTMSG	信息数据区	SCOLPDFN1H	XDS_SCCTMSG	SCCTMSG
SCCTBPA	业务平台数据区	SCOLPDFN1H	XDS_SCCTBPA	SCCTBPA

联机平台公用程序接口定义，源成员：SCOLPDFN1H，定义内容为：

数据名称	说明	公用定义源成员名称	公用定义数据结构名称	程序用数据结构名称
SDMSG	抛信息程序接口	SCOLPDFN1H	XDS_SCCSDMSG	SCCSDMSG
CLCPN	组件调用程序接口	SCOLPDFN1H	XDS_SCCCLCPN	SCCCLCPN
RPARM	取参数程序接口	SCOLPDFN1H	XDS_SCCRPARM	SCCRPARM
OUTPT	格式输出程序接口	SCOLPDFN1H	XDS_SCCOUTPT	SCCOUTPT
TRACE	跟踪调试程序接口	SCOLPDFN1H	XDS_SCCTRACE	SCCTRACE

EXCPT	例外处理程序接口	SCOLPDFN1H	XDS_SCCEXCPT	SCCEXCPT
BCCPN	组件调用前程序接口	SCOLPDFN1H	XDS_SCCCLCPN	SCCCLCPN
ACCPN	组件调用后程序接口	SCOLPDFN1H	XDS_SCCCLCPN	SCCCLCPN
MPARM	参数维护处理程序接口	SCOLPDFN1H	XDS_SCMPARM	SCCPARM

联机平台公用程序调用接口定义，源成员：SCOLPDFN1H，定义内容为：

公用程序名称	说明	公用定义源成员名称	调用名称	
SCSDMSGB	抛信息程序	SCOLPDFN1H	XPGM_SCSDMSG	
SCCLCPNB	组件调用程序	SCOLPDFN1H	XPGM_SCCLCPN	
SCRPARMB	取参数程序	SCOLPDFN1H	XPGM_SCRPARM	
SCOUTPTB	格式输出程序	SCOLPDFN1H	XPGM_SCOUTPT	
SCTRACEB	跟踪调试程序	SCOLPDFN1H	XPGM_SCTRACE	
SCEXCPTB	例外处理程序	SCOLPDFN1H	XPGM_SCEXCPT	
SCBCCPNB	组件调用前程序	SCOLPDFN1H	XPGM_SCBCCPN	
SCACCPNB	组件调用后程序	SCOLPDFN1H	XPGM_SCACCPN	
SCMPARMB	参数维护处理程序	SCOLPDFN1H	XPGM_SCMPARM	

程序的使用方法：

在 D 表使用 LIKEDS(公用数据结构名称)定义程序用数据结构名称；

在 D 表使用/COPY C@@DFNSRC,公用定义源成员名称；

在程序直接使用数据结构或数据结构的字段；

14.3 技术平台公用函数

为了方便 RPG 程序的开发，将技术平台的公用程序封装成公用函数，应用程序直接使用函数，不需要了解公用程序的调用方法。

联机平台公用函数的原型定义在公用定义源码中，源码库：CNCBSRCSCD，源文件：C@@DFNSRC，源成员：SCOLPSRV1H

联机平台公用函数清单：

公用程序	说明	公用函数原型定义源成员名称	公用函数名称	服务程序
SCSDMSGB	抛信息程序	SCOLPSRV1H	F_SCSDMSG	SCOLPSRV1

SCCLCPNB	组件调用程序	SCOLPSRV1H	F_SCCLCPN	SCOLPSRV1
SCRPARMB	取参数程序	SCOLPSRV1H	F_SCRPARM	SCOLPSRV1
SCOUTPTB	格式输出程序	SCOLPSRV1H	F_SCOUTPT	SCOLPSRV1
SCTRACEB	跟踪调试程序	SCOLPSRV1H	F_SCTRACE	SCOLPSRV1
SCBCCPNB	组件调用前程序	SCOLPSRV1H	F_SCBCCPN	SCOLPSRV1
SCACCPNB	组件调用后程序	SCOLPSRV1H	F_SCACCPN	SCOLPSRV1
SCMPARMB	参数维护程序	SCOLPSRV1H	F_SCMPARM	SCOLPSRV1

程序的使用方法：

在 D 表使用/COPY C@@DFNSRC,公用函数原型定义源成员名称；

在程序直接使用函数；

程序编译时联编服务程序；

14.4 技术平台公用子例程

为了方便 RPG 程序的开发，将技术平台的公用程序封装成公用子例程，应用程序直接执行公用子例程序，不需要了解公用程序的调用方法。

公用子例程序定义在公用定义源码中，源码库：CNCBSRCSCD，源文件：C@@DFNSRC，源成员：SCPUBPUBnR（n 为序号，1－9）；

公用子例程调用技术平台公用程序的原型定义在公用定义源码中。

技术平台公用子例程清单：

公用程序	说明	公用子例程定义源成员名称	公用子例程名称	调用程序公用定义源码
SCEXCPTB	例外处理程序	SCPUBSUB1R	*PSSR	SCOLPDFN1H

程序的使用方法：

在 D 表使用/COPY C@@DFNSRC,公用程序接口定义源成员名称；

在 D 表使用/COPY C@@DFNSRC,公用程序调用原型定义源成员名称；

在程序为公用程序接口赋值；

调用公用子例程；

在程序结束前，/COPY C@@DFNSRC,公用子例程定义源成员名称；

14.5 *PSSR 子例程的使用

为了统一进行 RPG 程序的异常处理，定义了公用的*PSSR 子例程，源码库：CNCBSRCD，源文件：C@@DFNSRC，源成员：SCPUBSUB1R。

*PSSR 需要使用程序反馈信息数据结构、文件反馈信息数据结构，系统统一定义了相应的数据结构，源码库：CNCBSRCD，源文件：C@@DFNSRC，源成员：@@DTADFN1H。

程序的使用方法：

在 F 表，对需要进行异常处理的文件，增加定义 INFDS(DS_FILE)、INFSR(*PSSR)；

在 D 表使用 LIKEDS(XDS_FILE)定义 DS_FILE；

在 D 表使用/COPY C@@DFNSRC,@@DTADFN1H；

在 D 表使用/COPY C@@DFNSRC,SCOLPDFN1H；

在程序结束前，使用/COPY C@@DFNSRC,SCPUBSUB1R；

14.6 技术平台公用函数、公用子例程的使用

14.6.1 抛信息处理（函数）

抛信息的调用方法：（红色部分为可选的）

```
        IF F_SCSDMSG(TWA 指针:信息码:字段序号:附加信息:继续标志:主控标志) =
G_TWA_RETF_Y ;
           EXSR @SR_RETURN ;
        ENDIF ;
```

最简单的抛信息方式，只抛出信息码：

```
        IF F_SCSDMSG(PTR_TWA: ‘SCR9901’ :*OMIT:*OMIT:*OMIT:*OMIT) =
```

```
G_TWA_RETF_Y ;
            EXSR @SR_RETURN ;
         ENDIF ;
```

增加字段序号、附加信息的抛信息方式：

```
         IF F_SCSDMSG(PTR_TWA:‘SCR9901’:2:’错误信息’:*OMIT:*OMIT) =
G_TWA_RETF_Y ;
            EXSR @SR_RETURN ;
         ENDIF ;
```

抛信息时同时发送主控方式：

```
         IF F_SCSDMSG(PTR_TWA:‘SCR9901’:*OMIT:*OMIT:*OMIT:G_SDMSG_CNSF_Y) =
G_TWA_RETF_Y ;
            EXSR @SR_RETURN ;
         ENDIF ;
```

如果需要信息发出后，仍然返回到应用程序，由应用程序控制处理流程，则需要置继续的控制标志。检查程序可以采用这种方式，一次性地检查出全部错误。程序代码如下：

```
         IF F_SCSDMSG(PTR_TWA:‘SCR9901’:*OMIT:*OMIT:G_SDMSG_CNTF_Y:*OMIT) =
G_TWA_RETF_Y ;
            EXSR @SR_RETURN ;
         ENDIF ;
```

14.6.2 组件调用处理（函数）

调用其他组件的方法：（红色部分为可选的）

```
         IF F_SCCLCPN(TWA 指针:组件名称:组件参数区地址：组件参数区长度:信息区地
址:信息序号:字段序号:继续标志:输出标志:错误控制标志) = G_TWA_RETF_Y ;
```

```
            EXSR @SR_RETURN ;
         ENDIF ;
```

继续标志、输出标志、错误控制标志的使用方法与子过程方式相同。

一般调用组件的方法：

```
         IF F_SCCLCPN(PTR_TWA: 'SCTEST1R' :%ADDR(SCTEST1):%LEN(SCTEST1):
*OMIT:*OMIT:*OMIT:*OMIT:*OMIT: *OMIT) = G_TWA_RETF_Y ;
            EXSR @SR_RETURN ;
         ENDIF ;
```

一般调用数据检查组件的方法：选择'继续标志'参数值为 G_CLCPN_CNTF_Y

```
         IF F_SCCLCPN(PTR_TWA: 'SCTEST2R' :%ADDR(SCTEST2):%LEN(SCTEST2):
*OMIT:*OMIT:*OMIT: G_CLCPN_CNTF_Y:*OMIT:*OMIT) = G_TWA_RETF_Y ;
            EXSR @SR_RETURN ;
         ENDIF ;
         IF SCCTWA.TWAEXCF = 'E' ;
            //错误处理流程
         ENDIF ;
```

由组件控制处理流程的调用方法：发生异常后，希望平台恢复到调用前的状态，控制权返回调用组件，则'错误控制标志'参数值为 G_CLCPN_HDLF_Y

```
         IF F_SCCLCPN(PTR_TWA: 'SCTEST2R' :%ADDR(SCTEST2):%LEN(SCTEST2):
*OMIT:*OMIT:*OMIT:*OMIT:*OMIT:G_CLCPN_HDLF_Y) = G_TWA_RETF_Y ;
            EXSR @SR_RETURN ;
         ENDIF ;
         IF SCCTWA.TWAMSCD <> *BLANK ;
            //错误处理流程
```

```
        ENDIF ;
```

14.6.3 取参数处理（函数）

```
        IF F_SCRPARM(TWA 指针:参数类型:参数代码:参数变量地址:参数变量长度:机构号) = G_TWA_RETF_Y ;
            EXSR @SR_RETURN ;
        ENDIF ;
        IF SCCTWA.TWADBRC = G_TWA_DBRC_NOM ;
            //成功处理流程;
        ELSE ;
            //异常处理流程;
        ENDIF ;
```

程序取多个参数的调用方法：

```
        IF F_SCRPARM(PTR_TWA:‘参数类型一’:’参数代码一’:%ADDR(参数变量一地址):%LEN(参数变量一长度):机构号) = G_TWA_RETF_Y ;
            EXSR @SR_RETURN ;
        ENDIF;
        IF SCCTWA.TWADBRC = G_TWA_DBRC_NOM ;
            //成功处理流程;
        ELSE ;
            //异常处理流程;
        ENDIF ;

        F_SCRPARM(PTR_TWA:‘参数类型二’:’参数代码二’:%ADDR(参数变量二地址):%LEN(参数变量二长度):机构号) = G_TWA_RETF_Y ;
            EXSR @SR_RETURN ;
        ENDIF ;
```

```
        IF SCCTWA.TWADBRC = G_TWA_DBRC_NOM ;
           //成功处理流程;
        ELSE ;
           //异常处理流程;
        ENDIF ;
```

14.6.4 格式输出处理（函数）

输出处理的调用方法：

```
        IF F_SCOUTPT(TWA 指针:格式名称:输出变量地址:输出变量长度:多页标志:结束标志) = G_TWA_RETF_Y ;
           EXSR @SR_RETURN ;
        ENDIF ;
```

单页输出的方式：

```
        IF F_SCOUTPT(PTR_TWA:’SCFMT01’:%ADDR(DS_SCOUT01):%LEN(DS_SCOUT01):*OMIT：C_ENDF) = G_TWA_RETF_Y ;
           EXSR @SR_RETURN ;
        ENDIF ;
```

多页输出的方式：

```
        IF F_SCOUTPT(PTR_TWA:’SCFMT02’:%ADDR(DS_SCOUT02):%LEN(DS_SCOUT012):G_OUTPT_LSTF_Y:C_ENDF) = G_TWA_RETF_Y ;
           EXSR @SR_RETURN ;
        ENDIF ;
```

如果返回的 C_ENDF 为‘Y’则表示达到输出记录数，不能再继续进行多页输出，一般情况下应该跳出循环。

14.6.5 跟踪调试处理（函数）

跟踪处理的调用方法：

```
F_SCTRACE(TWA 指针:输出方式:输出变量地址:输出变量长度:偏移量:输出内容)
```

以字符跟踪字段 SCCMSCD 的值：

```
F_SCTRACE(PTR_TWA:G_TRACE_FUNC_N:XC_SYPN:%ADDR(SCCMSCD):
%LEN(SCCMSCD:0:*OMIT);
```

以 16 制跟踪字段 SCCMSCD 的值：

```
F_SCTRACE(PTR_TWA:G_TRACE_FUNC_D:XC_SYPN:%ADDR(SCCMSCD):
%LEN(SCCMSCD:0:*OMIT);
```

以显示提示信息的方式：

```
F_SCTRACE(PTR_TWA:G_TRACE_FUNC_D:XC_SYPN:*OMIT:*OMIT:*OMIT:’开始处
理’);
```

14.6.6 组件调用前处理（函数）

调用组件前，调用此函数登记技术平台的信息：

```
IF F_SCBCCPN(TWA 指针:组件名称:组件参数区地址：组件参数区长度) =
G_TWA_RETF_Y ;
   EXSR @SR_RETURN ;
ENDIF ;
```

一般调用组件的方法：

```
IF F_SCBCCPN(PTR_TWA: ‘TTCHKMSTR’ :%ADDR(TTCHKMST):%LEN(TTCHKMST)
) = G_TWA_RETF_Y ;
```

```
            EXSR @SR_RETURN ;
        ENDIF ;
```

14.6.7 组件调用前处理（函数）

调用组件前，调用此函数登记技术平台的信息：

```
        IF F_SCACCPN(TWA 指针:组件名称:组件参数区地址：组件参数区长度) =
G_TWA_RETF_Y ;
            EXSR @SR_RETURN ;
        ENDIF ;
```

一般调用组件的方法：

```
        IF F_SCACCPN(PTR_TWA: ‘TTCHKMSTR’ :%ADDR(TTCHKMST):%LEN(TTCHKMST)
) = G_TWA_RETF_Y ;
            EXSR @SR_RETURN ;
        ENDIF ;
```

14.6.8 参数维护处理（函数）

```
        IF F_SCMPARM(TWA 指针:参数指针:操作方式:返回码) = G_TWA_RETF_Y ;
            EXSR @SR_RETURN ;
        ENDIF ;
        IF 返回码 = G_MGID_OK ;
            //成功处理流程;
        ELSE ;
            //异常处理流程;
        ENDIF ;
```

其中：

参数指针指向存放内容的参数数据结构，参数结构结构应该与文件 SCPRMA 结构相同。银行号、参数类型、参数代码、生效日期、失效日期等字段内容必须输入。

操作方式为 1 位字符，必须输入：‘C’建立、‘U’更新、‘D’删除、‘Q’查询、‘S’开始浏览、‘N’读下一笔、‘E’结束浏览；

返回码为 7 位字符，由函数返回；

调用方法：（查询方式）

```
(参数关键字赋值)
IF F_SCMPARM(PTR_TWA:%ADDR(DS_SCPRM):’Q’:C_MGID) = G_TWA_RETF_Y ;
   EXSR @SR_RETURN ;
ENDIF;
IF C_MGID = G_MGID_OK ;
   //成功处理流程;
ELSE ;
   //异常处理流程;
ENDIF ;
```

14.6.9 异常处理（子例程）

异常处理一般用于捕获程序的异常错误（系统错误）。

在 RPG 程序中使用*PSSR 捕获错误，然后调用异常组件。

```
*****************************************************************
* 过程名称：*PSSR
* 功能描述：异常处理子过程
*****************************************************************
/FREE
   BEGSR *PSSR ;
      SCCEXCPT.EXCPTTYPE = G_EXCPT_TYPE_D ;
```

```
            SCCEXCPT.EXCPTPROG = XC_SYPN ;
            SCCEXCPT.EXCPTSRCL = %DEC(XC_LINE:8:0) ;
            SCCEXCPT.EXCPTFUNC = DS_FILE.XC_OPCD ;
            SCCEXCPT.EXCPTOBNM = XC_FILE ;
            CALL(P) XPGM_SCEXCPT(SCCTWA:SCCEXCPT) ;
        ENDSR ;
    /END-FREE
```

异常处理与*PSSR 子例程结合使用，不需要程序增加处理。

系统发生异常时自动落入*PSSR 中。只需要使用/COPY 方式复制*PSSR 相关的公用定义即可。参见《*PSSR 子例程序的使用》一节的内容。

14.7 技术平台输出区（TOA）的说明

TOA 的私有域由两部分组成：标准化输出区、格式化输出区。

标准化输出区包括：交易组件输出（zznnnnES 结构的内容，或zznnnnES+zznnnnEM），多页输出的内容（zznnnnEL）

格式化输出区包括：信息输出、回单输出

14.7.1 标准化输出区说明

如果 TOA 的字段 O_SDPL>0，则表示标准化输出区内有数据。

14.7.1.1 只包含组件输出时结构说明

输出内容说明：

组件输出长度 5 个字节	格式码 10 个字节	输出内容			
Nnnnn 格式码长度（10）+组件输出格式长度（zznnnnE 长度）	zzOnnnnn 2 位应用子模块+‘0’+7 位交易码	输出内容 与数据结构 zznnnnE 匹配			

0_SDPL=组件输出长度+5=5+10（格式码长度）+zznnnnE 结构长度

zznnnnE 是由 zznnnnES+zznnnnEM 组成，也只按 zznnnnE 返回内容，由应用解析为 zznnnnES、zznnnnEM。

14.7.1.2 只包含多页查询输出时结构说明

输出内容说明：

多页输出长度 5 个字节	格式码 10 个字节	多页输出内容			
Nnnnn 格式码长度（10）+多页查询输出格式长度（zznnnnEL 长度）*返回记录数	zzLnnnnn 2 位应用子模块+‘L’+7 位交易码	输出内容 与数据结构 zznnnnEL*返回记录数匹配			

0_SDPL=组件输出长度+5=5+10（格式码长度）+zznnnnEL 结构长度*返回记录数

返回多条记录时，也只返回一个格式码 zzLnnnnnn，需要应用按记录数解析为 zznnnnEL。

14.7.1.3 同时返回组件输出及多页查询时结构说明

输出内容说明：此时标准化输出区包含两个格式码。

组件输出长度 5 个字节	格式码 10 个字节	输出内容	多页输出长度 5 个字节	格式码 10 个字节	输出内容
Nnnnn 格式码长度（10）+组件输出格式长度（zznnnnE 长度）	zzOnnnnn 2 位应用子模块+‘0’+7 位交易码	输出内容 与数据结构 zznnnnE 匹配	Nnnnn 格式码长度（10）+多页查询输出格式长度（zznnnnEL 长度）*返回记录数	zzLnnnnn 2 位应用子模块+‘L’+7 位交易码	输出内容 与数据结构 zznnnnEL*返回记录数匹配

0_SDPL=组件输出长度+5+多页输出长度+5=5+10（格式码长度）+zznnnnE 结构长度+5+10（格式码长度）+zznnnnEL 结构长度*返回记录数

zznnnnE 是由 zznnnnES+zznnnnEM 组成，也只按 zznnnnE 返回内容，由应用解析为 zznnnnES、zznnnnEM。

返回多条记录时，也只返回一个格式码 zzLnnnnnn，需要应用按记录数解析为 zznnnnEL。

多页查询时，翻页处理后的标准化输出区只包含多页查询格式，不再包含组件输出格式。

14.7.2 格式化输出区说明

如果 TOA 的字段 O_OUPL>0，则表示格式化输出区内有数据。

14.7.2.1 信息输出区结构说明

格式化输出区可以输出多条信息，每一条信息都是带有长度、格式码、信息内容：

信息输出格式长度 5 个字节	信息格式码 10 个字节	信息码 7 个字节	附加信息长度 7 个字节	附加信息	
Nnnnn	SCOPMS0001	信息码		如果附信息长度=0，则没有这此内容	

信息输出格式长度=5+10（格式码长度）+7（信息码长度）+7（附加信息长度字段）+附加信息长度字段

如果有多条信息，则在格式化输出区顺序排列。

如果格式化输出区只输出信息，则格式化输出区的长度为：

格式化输出区长度=5+信息输出格式长度 1+5+信息输出格式长度 2+…+5+信息输出格式长度 n

由于每组信息输出格式不固定（附加信息长度不固定），所以必须按信息输出格式实际长度进行解析。

14.7.2.2 回单输出区结构说明

暂无。

14.8 预编译语法

14.8.1 预编译基本语法说明

定义内容	预编译语法	说明
预编译开始	//@_@PRCPL ‘预编译功能标识’ 例： //@_@PRCPL SNDMSG	必须以//PRCPL 开始 预编译功能标识与//PRCPL 之间必须保留至少一个空格 目前支持的预编译功能标识： SNDMSG（发送信息） CALBEF（单元/函数组件调用前处理） CALAFT（单元/函数组件调用后处理） CALBSV（服务组件调用前处理） CALASV（服务组件调用后处理） OUTPUT（多页输出）
预编译结束	//@_@END-PRCPL	必须为//END-PRCPL

预编译参数定义	// 参数名称(参数值)
	或
	// 参数名称
	
	例：
	// PGM(TTGETMST)
	
	// CONTINUE
		必须以//开始，参数名称与//之间必须保留至少一个空格
		参数名称必须与配置中的参数标识一致
		目前支持的参数名称：
		PGM：组件名称，有参数值
		FIELD：字段序号，有参数值
		CONTINUE：继续标志，无参数值
		CONSOLE：主控标志，无参数值

预编译省略语	//@_@PRCPL ‘预编译功能标识’ ‘组件名称’ 例： //@_@PRCPL SNDMSG TTGETMST	必须以//PRCPL 开始 预编译功能标识与//PRCPL 之间必须保留至少一个空格 组件名称与预编译功能标识之间必须保留至少一空格

14.8.2 预编译基本功能语法

预编译功能	预编译功能语法	说明
CLABEF	标准写法： //@_@PRCPL CALBEF // PGM(组件名称) //@_@END-PRCPL 例：	单元/函数组件调用前处理 无可以省略项

	//@_@PRCPL CALBEF // PGM(TTGETMST) //@_@END-PRCPL	
	省略写法： //@_@PRCPL CALBEF 组件名称 例： //@_@PRCPL CALBEF TTGETMST	

CALAFT	标准与法：
	//@_@PRCPL CALAFT
	// PGM(组件名称)
	//@_@END-PRCPL
	
	例：
	//@_@PRCPL CALAFT
	// PGM(TTGETMST)
	//@_@END-PRCPL
		单元/函数组件调用后处理
		无可以省略项

	省略写法： //@_@PRCPL CALAFT 组件名称 例： //@_@PRCPL CALAFT TTGETMST	

CLABSV	标准写法：
	//@_@PRCPL CALBSV
	// PGM(组件名称)
	//@_@END-PRCPL
	
	例：
	//@_@PRCPL CALBSV
	// PGM(DD1103)
	//@_@END-PRCPL
		服务组件调用前处理
		无可以省略项

	省略写法： //@_@PRCPL CALBSV 组件名称 例： //@_@PRCPL CALBSV DD1103	
CALASV	标准与法： //@_@PRCPL CALASV // PGM(组件名称) //@_@END-PRCPL 例：	服务组件调用后处理 无可以省略项

	//@_@PRCPL CALASV // PGM(DD1103) //@_@END-PRCPL	
	省略写法： //@_@PRCPL CALASV 组件名称 例： //@_@PRCPL CALASV DD1103	

SNDMSG	//@_@PRCPL SNDMSG
	// PGM(组件名称)
	// FIELD(字段名称)
	// CONTINUE
	// CONSOLE
	//@_@END-PRCPL
	
	例：
	//@_@PRCPL SNDMSG
	// PGM(TTGETMST)
	//@_@END-PRCPL
		发送消息处理
		组件名称必须定义，可以是*PGM。
		组件名称不是*PGM：则系统自动以’组件名称 ES.ESRTCD’、’组件名称 ES.ESAINF’作为信息码、附加信息
		组件名称是*PGM：则系统自动以’XC_MGID’、’XC_AINF’作为信息码、附加信息
		
		字段名称可选
		CONTINUE、CONSOLE 可选

	省略写法： //@_@PRCPL SNDMSG 组件名称 例： //@_@PRCPL SNDMSG TTGETMST 或 //@_@PRCPL SNDMSG *PGM	发送函数返回的信息时，需要指定组件名称； 发送本组件的信息时，组件名称为*PGM，组件必须为公用变量 XC_MGID、XC_AINF 赋值。

OUTPUT	标准与法：
	//@_@PRCPL OUTPUT
	// PGM(组件名称)
	//@_@END-PRCPL
	
	例：
	//@_@PRCPL OUTPUT
	// PGM(TTGETMST)
	//@_@END-PRCPL
		格式化输出时使用，目前只支持多页输出方式。

	省略写法： //@_@PRCPL OUTPUT 组件名称 例： //@_@PRCPL OUTPUT TT3500	

14.8.3 发送消息功能的使用

程序功能	预编译功能语法	说明

数据检查过程中发送消息	标准写法：
	//@_@PRCPL SNDMSG
	// PGM(组件名称)
	// CONTINUE
	//@_@END-PRCPL
	
	例：
	调用函数组件 TTGETMST 后发送信息，并希望继续处理
	//@_@PRCPL SNDMSG
	// PGM(TTGETMST)
	// CONTINUE
	//@_@END-PRCPL
	
	组件发送信息，并希望继续处理
	//@_@PRCPL SNDMSG
	// PGM(*PGM)
	// CONTINUE
	//@_@END-PRCPL
		为了完成全部检查功能，调用检查组件后，如果有错误，发送消息，但不希望结束程序，可以使用参数 CONTINUE 方式
		
		使用此种方式，不支持省略写法
		
		全部检查处理执行后，必须检查 TWA 的信息状态，如果异常则应该结束程序，代码如下：
		IF SCCTWA.I_MSGT = G_TWA_MSGT_ERR ;
		EXSR @SR_RETURN ;
		ENDIF ;

数据处理过程中发送消息	标准与法：
	//@_@PRCPL SNDMSG
	// PGM(组件名称)
	//@_@END-PRCPL
	
	例：
	//@_@PRCPL SNDMSG
	// PGM(TTGETMST)
	//@_@END-PRCPL
	或
	//@_@PRCPL SNDMSG
	// PGM(*PGM)
	//@_@END-PRCPL
		如果数据处理中发生错误，发送消息后应该结束处理，不能使用参数 CONTINUE。

	省略写法： //@_@PRCPL SNDMSG 组件名称 例： //@_@PRCPL SNDMSG TTGETMST 或 //@_@PRCPL SNDMSG *PGM	

各位同事，大家好！

RPG 编码规范经过近期实践，少量修订。与 V0.65 相比较，修订了以下内容：

此次开发规范修改的内

容:**********

4.1.4 批处理组件命名规则(新增内容)

4.3.7 多页查询使用外部文件的结构命名(新增内容,说明多页查询使用文件结构的方法)

6.2.4 交易服务组件接口数据结构的定义(修改了循环结构的定义,增加 DIM 定义)

6.2.5 其他服务组件接口数据结构的定义(修改了循环结构的定义,增加 DIM 定义)

6.2.6 单元组件\函数组件接口数据结构的定义(修改了循环结构的定义,增加 DIM 定义)

14.8.2 预编译基本功能语法(增加回单的输出方法定义 OUTFMT)