**ERP项目开发规范**

小 组：开发组

提交日期：2014/12/20

当前版本：V1.0

**文档控制**

**文档更新记录**

| **日期** | **更新人** | **版本** | **备注** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**文档审核记录**

| **日期** | **审核人** | **职务** | **备注** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**目录**

1 ABAP对象命名规范 - 1 -

1.1 开发类 - 1 -

1.2 程序名 - 1 -

1.3 数据字典 - 2 -

1.3.1 表/结构/数据元素/域 - 2 -

1.3.2 表类型 - 2 -

1.3.3 范围表类型 - 2 -

1.3.4 搜索帮助 - 2 -

1.3.5 锁对象 - 2 -

1.4 函数组及函数模块 - 2 -

1.4.1 函数组 - 2 -

1.4.2 函数模块 - 3 -

1.5 BADI实施 - 3 -

1.6 消息类 - 3 -

1.7 类 - 4 -

1.8 SMARTFORM - 4 -

1.9 自定义权限对象 - 4 -

1.10 Enhancement project - 4 -

1.11 Web Service命名 - 4 -

1.12 Proxy 命名 - 4 -

2 通用ABAP代码规范 - 5 -

2.1 代码格式 - 5 -

2.1.1 Pretty Printer - 5 -

2.1.2 Spacing - 5 -

2.1.3 代码行 - 6 -

2.1.4 代码注释 - 7 -

2.2 程序变量 - 8 -

2.2.1 变量使用说明 - 8 -

2.2.2 命名规范 - 8 -

2.2.3 常量 - 9 -

2.2.4 全局变量 - 11 -

2.2.5 内表/结构 - 12 -

2.2.6 文本变量 - 14 -

2.3 常用语句格式 - 14 -

2.3.1 Message - 14 -

2.3.2 Call function - 15 -

2.3.3 Perform - 15 -

2.3.4 Select - 16 -

2.3.5 IF - 17 -

2.3.6 Check - 17 -

2.3.7 Loop - 18 -

2.3.8 Read - 18 -

2.3.9 While - 19 -

2.3.10 Do - 19 -

2.3.11 Case - 19 -

2.3.12 Concatenate - 21 -

3 报表程序规范 - 22 -

3.1 程序结构 - 22 -

3.1.1 数据初始化子程序 - 24 -

3.1.2 数据存取子程序 - 25 -

3.1.3 数据处理子程序 - 26 -

3.1.4 数据组合子程序 - 26 -

3.1.5 数据输出子程序 - 28 -

4 子程序规范 - 29 -

4.1 模块化代码 - 29 -

4.2 参数 - 30 -

4.2.1 参数命名 - 30 -

4.2.2 参数类型 - 30 -

4.2.3 Table参数 - 31 -

4.2.4 参数个数要求 - 31 -

4.2.5 参数对齐 - 33 -

4.3 子程序结构 - 33 -

4.4 Perform语句 - 35 -

4.5 子程序注释 - 35 -

5 功能模块规范 - 36 -

5.1 函数模块结构 - 36 -

5.2 函数返回值 - 36 -

5.3 函数调用 - 36 -

6 屏幕对话程序规范 - 38 -

6.1 屏幕逻辑流结构 - 38 -

6.2 PAI/PBO/POV 模块 - 39 -

6.2.1 User-command模块 - 40 -

7 BADI增强规范 - 42 -

8 CMOD增强规范 - 44 -

9 BTE增强规范 - 45 -

10 程序性能相关代码规范 - 46 -

10.1 SQL语句 - 46 -

10.2 内表操作 - 51 -

10.3 其他语句 - 58 -

10.3.1 CASE语句 - 58 -

10.3.2 WHILE语句 - 59 -

10.3.3 比较语句 - 60 -

附录 - 61 -

1.1 SAP业务模块命名表 - 61 -

1.2 数据字典对象类型命名表 - 61 -

1.3 程序类型命名表 - 61 -

# ABAP对象命名规范

## 开发类

Z<XX>\_<Short text>

<XX>－业务模块，见附录1

<Short text>－简短描述，英文

## 程序名

Z<XXY>\_<NNNN>

<XX>－业务模块，见附录1

<Y>－程序类型代码

<NNNN>-开发需求编号，为四位数字

|  |  |
| --- | --- |
| <Y> | 程序类型代码 |
| R | 报表程序 |
| P | 对话程序 |
| F | FORM/SMARTFORM打印程序 |
| C | 数据转换程序） |
| A | 组件程序 |
| I | 公共Include程序 |

注：用于特定程序的include程序按以下方式命名：<主程序名>\_<XNN>, NN-两位流水码

|  |  |
| --- | --- |
| <X> | include类型 |
| F | 子程序 |
| I | PAI |
| O | PBO |

## 数据字典

### 表/结构/数据元素/域

Z<TXX>\_<Short text>

<T>－数据字典对象类型，参见附表2

<XX>－业务模块，见附录1

<Short text>－简短描述，英文

### 表类型

[Z\_]<structure >\_T

[Z\_]－如果< structure >名称以Z开头，则可省略

<structure>－使用的结构名

### 范围表类型

[Z\_]<dataelement>\_T

[Z\_]－如果<dataelement>名称以Z开头，则可省略

<dataelement>－使用的数据元素名

### 搜索帮助

Z<XX>\_SH\_<Short text>

<XX>－业务模块，见附录1

<Short text>－简短描述，英文

### 锁对象

E[Z] <tablename>

[Z]－如果<tablename>名称以Z开头，则可省略

<tablename>－表名

## 函数组及函数模块

### 函数组

Z<XX>\_<nn>

<XX>－业务模块，见附录1

<nn>－两位流水码

### 函数模块

<FunctionGroup>\_<Short text>

<FunctionGroup>－函数模块所在的函数组名

< Short text>－函数基本功能的短文本

## BADI实施

a) Custom Business Add-in

Z<XX>\_BADI\_< Short text>

<XX>－业务模块，见附录1

< Short text>－BADI 定义的名称

b) Custom Business Add-in Implementation

Z<XX>\_BADIM\_< Short text>

<XX>－业务模块，见附录1

<Short text>－BADI 定义的名称

c) SAP Business Add-in Implementation

Z<XX>\_IMPL\_< Short text>

<XX>－业务模块，见附录1

<Short text>－BADI 定义的名称

d) SAP BADI Method

< Short text>－自由定义,但有意义,简单描述该方法实现的功能

## 消息类

Z<XX>\_<nn>

<XX>－业务模块，见附录1

< Short text>－简短描述，英文

## 类

ZC<XX>\_<Short text>

<XX>－业务模块，见附录1

<Short text>－简短描述，英文

## SMARTFORM

[Z]<Reportname>\_<Short text>

[Z]－如果<Reportname>名称以Z开头，则可省略

<Short text>－简短描述，英文

## 自定义权限对象

Z<XX>\_<Short text>

<XX>－业务模块，见附录1

<Short text>－简短描述，英文

## Enhancement project

Z<XX><nnnn>

<XX>－业务模块，见附录1

<nnnn>－四位流水码

## Web Service命名

ZWS<XX>\_<Short text>

<XX>－业务模块，见附录1

<Short text>－简短描述，英文

## Proxy 命名

Z<XX>\_<Short text>

<XX>－业务模块，见附录1

<Short text>－简短描述，英文

# 通用ABAP代码规范

## 代码格式

### Pretty Printer

所有程序代码应使用pretty printer功能进行规范，包含：代码缩进；关键字大写，其余小写。

***Unacceptable*:**

Form frm\_list\_output.

data: wa\_output type ty\_output.

if i\_output[] is initial.

message e101(zcs01).

endif.

loop at i\_output into wa\_output.

write: / wa\_output-equnr.

endloop.

endform.

***Acceptable*:**

Form frm\_list\_output.  
  DATA: wa\_output TYPE ty\_output.  "work area for output  
  
  IF i\_output[] IS INITIAL.  
    MESSAGE e101(zcs01). "error no data  
  ENDIF.  
  
  LOOP AT i\_output INTO wa\_output.  
    WRITE: / wa\_output-equnr.  
  ENDLOOP.  
ENDFORM.                    "f\_list\_output

### Spacing

在代码行中按照代码块适当增加空行，以增加代码的可读性：

***Unacceptable:***

form sub\_display\_print\_document tables pi\_ropbel type ty\_tab\_opbel.

data: l\_type type c. “print type

data: lw\_opbel type ty\_opbel. “Loop work area

l\_type = c\_print\_document.

format color col\_blue on.

write: / text-h01. “'Print documents (click to view)'.

format color off.

if not pi\_ropbel[] is initial.

loop at pi\_opbel\_disp into lw\_opbel\_disp.

write: / lw\_opbel-partner.

endloop.

endif.

clear l\_type.

endform. " display\_print\_document

***Acceptable:***

FORM sub\_display\_print\_document TABLES pi\_ropbel TYPE ty\_tab\_opbel.  
  
  DATA: l\_type   TYPE c.      "print type  
  DATA: lw\_opbel TYPE ty\_opbel.   "Loop work area  
  
  l\_type = c\_print\_document.  
  
  FORMAT COLOR COL\_GROUP ON.  
  WRITE: / text-h01.  "'Print documents (click to view)'.  
  FORMAT COLOR OFF.  
  
  IF NOT pi\_ropbel[] IS INITIAL.  
    LOOP AT pi\_opbel\_disp INTO lw\_opbel\_disp.  
      WRITE: / lw\_opbel-partner.  
    ENDLOOP.  
  ENDIF.  
  
  CLEAR l\_type.  
ENDFORM.                    " display\_print\_document

### 代码行

每一行代码仅允许有一个语句

***Unacceptable*:**

Sort i\_tab1 by contract. Sort i\_tab2 by anlage.

Clear: lwa\_tab1, lwa\_tab2.

***Acceptable*:**

Sort i\_tab1 by contract.

Sort i\_tab2 by anlage.

Clear: lwa\_tab1, “Clear work areas for processing

　　 lwa\_tab2. “them in the loop.

### 代码注释

代码行中的注释，应该与代码行对齐。

***Unacceptable*:**

Form frm\_output\_data.

\*Print the headings

perForm frm\_write\_column\_heading.

\*\*\* Loop at output table

Loop at i\_output into w\_output.

\*Check if the document is reverse.

If w\_output-reverse = c\_reversed “Reversed state

and w\_output-simulated = c\_unsimluated. ”simulated

Endif.

Endloop.

Endform.

***Acceptable*:**

Form frm\_output\_data.

\* Print the headings

perForm frm\_write\_column\_heading.

\* Loop at output table

Loop at i\_output into w\_output.

\* Check if the document is reverse.

If w\_output-reverse = c\_reversed “Reversed state

and w\_output-simulated = c\_unsimluated. ”simulated

Endif.

Endloop.

Endform.

## 程序变量

### 变量使用说明

基本原则：程序中尽量减少全局变量的使用，如无必要均应使用局部变量，以保证程序中各代码段的相对独立性。全局变量一般用于：跨子程序使用的内表、常量、屏幕变量等。

变量声明：局部变量声明语句必需放在子程序的开始位置；不允许在一行代码中定义两个或以上的变量；在同一个data语句中的所有变量，应该让变量名对齐、type〔like〕对齐。

变量使用：原则上每个变量仅为一种用途使用，尽量避免一个变量被当作不同的含义重复使用。

***Unacceptable*:**

data: l\_cnt type i, l\_txt\_prefix(10) type c.

data　: l\_curr\_div like eanl-sparte.

***Acceptable*:**

data: l\_prefix(3)　 type c. “Prefix of operand

l\_cnt 　 type i, “Count of material docs.

l\_curr\_div like eanl-sparte. “Current division

### 命名规范

* 变量：<A><B>\_<description>

<A>－代表变量的使用范围

<B>－代表变量类别

|  |  |
| --- | --- |
| <A> | 使用范围 |
| G | 全局变量 |
| L | 局部变量 |
| C | 全局常量 |
| I | 形式参数－传入 |
| E | 形式参数－传出 |

|  |  |
| --- | --- |
| <B> | 变量类别 |
| V | 简单变量 |
| T | 内表变量 |
| S | 结构变量 |
| R | range |

* 类型：TY\_<description>

***Example:***

TYPES: BEGIN OF ty\_po\_header,

Ebeln TYPE ekko-ebeln, “PO NUMBER

Lifnr TYPE ekko-lifnr, “VENDOR ID

END OF ty\_po\_header.

CONSTANTS: BEGIN OF cs\_po\_type,

normal TYPE ekko-bsart VALUE ‘PO1’,

special TYPE ekko-bsart VALUE ‘PO2’,

END OF cs\_po\_type.

DATA: gt\_po\_header TYPE TABLE OF ty\_po\_header,

Gs\_po\_header TYPE ty\_po\_header.

DATA: gf\_error(1) TYPE c,

Gv\_datum TYPE sy-datum.

### 常量

#### 常量组

如果程序中定义了一组相关的常量（如下例），必须要要注意常量的命名。

***Example:***

Constants:

C\_periodic\_mr\_reason like eablg-ablesgr value ‘01’,

C\_check\_mr\_reason like eablg-ablesgr value ‘10’.

***(以上命名方式是不符合要求的)***

常量组的命名方式必须采用以下两种方式之一：

* 定义为结构，结构中包含所有相关常量：

***Example:***

constants: begin of cx\_mrr

periodic like eablg-ablesgr value ‘01’, “Periodic

check like eablg-ablesgr value ‘10’, “Check

End of cx\_mrr.

*这种常量的使用和普通的结构用法相同：*

*lwa\_eablg-ablesgr = cx\_mrr-periodic*

* 使用相同的前缀：

***Example:***

constants:

**c\_mrr**\_periodic like eablg-ablesgr value ‘01’,

**c\_mrr**\_check like eablg-ablesgr value ‘10’.

constants:

**c\_fi\_doc**\_inv(2) type c value ‘IN’,

**c\_fi\_doc**\_secdep(2) type c value ‘SD’.

#### 常量名称

常量命名时不应该使用常量值作为其名称的一部分，而应该使用常量值所代表的业务含义或者是其在程序中的用途。

***Unacceptable*:**

constants:

c\_9(2) type c value ‘9’,

c\_one(1) type c value ‘1’.

***Acceptable*:**

constants:

c\_status\_cleared(2) type c value ‘9’,

c\_status\_active(1) type c value ‘1’.

#### 常量声明

所有的常量均应定义为程序中的全局变量，不允许在子程序中定义局部常量。

### 全局变量

#### 最小化全局变量的使用

为了使程序代码模块化，降低各模块之间的耦合度，必须尽量减少程序中使用的全局变量，如无必要尽量使用局部变量。

以下类型的变量可以定义为全局变量：

* 内表（跨子程序使用）
* 计数器
* 标识
* 常量**（必须定义为全局）**
* 结构
* 范围变量
* 类型定义**（必须定义为全局）**

以下类型的变量不能定义为全局变量：

* 工作区（如果需要，工作区应该定义为子程序参数进行传递）

#### 变量组定义为结构

如果需要使用一组相关的全局变量，建议定义成一个全局的结构，而不是分别定义每个变量。

***Unacceptable*:**

data: g\_cnt\_billdocs type i,

g\_cnt\_invoices type i,

g\_cnt\_contracts type i.

***Acceptable*:**

types: begin of ty\_cnt,

billdocs type i, “Number of billing documents

invoices type i, “Number of unsimulated invoices

contracts type i. “Number of related contracts

end of ty\_cnt.

...

data: x\_cnt type ty\_cnt. “Structure to keep track of counters

#### 使用包含（include）

如果程序中定义全局变量的代码超过100行（包括全局变量、类型、常量以及相关的注释），必须单独建立一个include，将所有的变量声明语句从主程序中剥离出来，放到该包含程序中。

### 内表/结构

#### 内表定义

* 使用类型声明内表

内表的定义必须使用表类型。

***Unacceptable*:**

data: begin of i\_mrtab occurs 0,

ablbelnr like eabl-ablbelnr,

adatsoll like eabl-adatsoll,

ablstat like eabl-ablstat,

end of i\_mrtab.

***Acceptable*:**

types: begin of ty\_mrtab,

ablbelnr like eabl-ablbelnr, “MR Doc number

adatsoll like eabl-adatsoll, “Schedule MR Date

ablstat like eabl-ablstat, “MR Status

end of ty\_mrtab.

types: ty\_tab\_mrtab type [tablekind] table of ty\_mrtab.

...

data: i\_mrtab type ty\_tab\_mrtab.

* 对齐与注释

结构类型中定义的字段必须要按以下格式对齐；如果使用标准表字段，则字段名应该和标准表字段一致；结构类型中的字段后都应该有相应的文本注释。

***Unacceptable*:**

types: begin of ty\_mrtab,

mrdoc like eabl-ablbelnr,

date like eabl-adatsoll,

ablstat like eabl-ablstat,

end of ty\_mrtab.

***Acceptable*:**

（注意：字段对齐，字段名称与表字段一致，字段后应该有文本注释）

types: begin of ty\_mrtab,

ablbelnr like eabl-ablbelnr, **“MR Doc number**

adatsoll like eabl-adatsoll, **“Schedule MR Date**

ablstat like eabl-ablstat, **“MR Status**

end of ty\_mrtab.

#### 抬头行（Header Line）

定义内表时**不能**使用**with header line**定义其默认工作区，工作区必须在各个子程序中单独定义为局部变量。

***Unacceptable*:**

data: i\_mrtab type ty\_tab\_mrtab with header line.

***Acceptable*:**

Form frm\_process\_mr\_data.

**data: lwa\_mr type ty\_mrtab.**  “work area for i\_mrtab

loop at i\_mrtab into lwa\_mr.

...

endloop.

Endform.

#### 工作区

工作区不能定义为全局变量而应该在各个子程序中定义为局部变量使用。如果必须在子程序间传递工作区内容，则应该为相关子程序定义形式参数来实现。

Form frm\_process\_mr\_data.

**data: lwa\_mr type ty\_mrtab.**  “work area for i\_mrtab

loop at i\_mrtab into lwa\_mr.

...

perForm frm\_validate\_mr using **lwa\_mr.**

...

endloop.

Endform.

### 文本变量

所有文本均应定位为文本元素（text element）；如果不是必须，不允许在程序中使用文本硬代码或者将文本定义为常量来使用。

***Unacceptable*:**

write: ‘Purchase’.

***Acceptable*:**

write: text-c01. “Purchase

定位文本元素时应该使用合适的前缀；以下作为参考，可根据实际需要进行扩展。

|  |  |  |
| --- | --- | --- |
| **Prefix** | **Description** | **Example** |
| H | Header text | text-H01 |
| C | Column heading | text-C01 |
| E | Error text | text-E08 |
| W | Warning text | text-W03 |
| I | Information text | text-I19 |
| S | Summary text | text-S02 |
|  |  |  |

## 常用语句格式

### Message

Message语句应该使用以下格式:

Message <message type>(<message class>) with <variable 1><variable 2> etc.

***Example:***

message e304(ZCS01) with w\_data-anlage w\_data-device.

### Call function

函数调用语句中，如果有不使用的参数，请将这些行删除而不要将这些行注释在代码中。同时，请注意所有的参数必须在代码中对齐。

***Unacceptable*:**

CALL FUNCTION 'BAPI\_IDENTIFICATIONDETAILS\_GET'

EXPORTING

BUSINESSPARTNER = W\_EBILL\_CA-GPART

TABLES

\* IDENTIFICATIONDETAIL = I\_BAPIBUS1006\_ID\_DETAILS

RETURN = I\_BAPIRET2.

***Acceptable*:**

CALL FUNCTION 'BAPI\_IDENTIFICATIONDETAILS\_GET'

EXPORTING

BUSINESSPARTNER = W\_EBILL\_CA-GPART

TABLES

RETURN = I\_BAPIRET2.

*(请注意，这里的所有参数都工整地对齐)*

### Perform

使用perform语句调用子程序时，参数必须要对齐，且每一个参数单独一行，同时perform 语句及使用的参数必须有相应的注释。

***Unacceptable*:**

perForm frm\_delete\_unused\_bill\_orders tables i\_billorders using l\_date.

***Acceptable*:**

\* Use the standard function to delete unused billing orders from the list

\* in I\_BILLORDERS

perForm frm\_delete\_unused\_bill\_orders tables i\_billorders “Bill order list

using l\_date. “Input date

### Select

* Select 语句结构

Select语句必须使用以下示例中的结构：

**Example 1:**

SELECT logiknr   "logical device no.  
     equnr     "equipment number  
     bis      "to date  
    INTO wa\_dev  
    FROM egerh  
   WHERE equnr EQ l\_equip\_no     "equipment number  
    AND bis  GE p\_rundate      "date ranges are in  
    AND ab   LE p\_rundate.     "todays date.

* + 每一个字段必须分行列出，且对齐
  + 每一个选取的字段必须有注释
  + “AND” 关键字必须分行列出，且对齐
  + “EQ”, “GE” 等操作符必须对齐
  + 不要使用“NE”, “EQ”, “GT’, “LT”等操作符, 而应该使用“<>”, “=”等相同含义的操作符

**Example2:**

SELECT ekpo~ebeln  "po number  
     ekpo~ebelp  "po item  
     ekbe~belnr  "invoice  
    INTO TABLE gt\_ekbe  
    FROM ekko  
   INNER JOIN ekpo  
      ON ekko~ebeln EQ ekpo~ebeln  
   INNER JOIN ekbe  
      ON ekpo~ebeln EQ ekbe~ebeln  
     AND ekpo~ebelp EQ ekbe~ebelp  
   WHERE ekko~lifnr EQ iv\_lifnr               "Vendor Number  
     AND ( ekko~bsart EQ cs\_po\_type-normal OR "PO Normal type  
           ekko~bsart EQ cs\_po\_type-special ) "PO Special type  
     AND ekbe~vgabe EQ cv\_vgabe\_invoice.      "invoice type

* + 一般情况下不要使用表的别名，除非某个表在select语句中使用了两次以上
  + Inner joins 必须按以上格式对齐
  + Select语句中的关键字必须按以上格式对齐
  + 语句中的复杂条件语句必须按以上格式书写
* Select语句返回值

每一个select语句后面必须有相应的语句检查状态变量SY-SUBRC。如果确实不需要检查，那么必须有相应的注释说明原因。

### IF

IF语句的结构和对齐方式如下例：

**Example:**

  IF w\_tab-field1 EQ l\_field1 AND  
    w\_tab-field2 EQ l\_field2 OR  
     ( w\_tab-field3 LE l\_field3 AND  
       w\_tab-field4 GE l\_field4 ).  
    ...  
  ENDIF.

(请注意语句中变量、操作符以及AND/OR的对齐方式)

### Check

CHECK语句的结构和对齐方式如下例：

**Example:**

CHECK w\_tab1 IS NOT INITIAL AND  
     w\_tab2 IS NOT INITIAL.

在可能的情况下，尽量使用CHECK语句从而避免IF语句的多层嵌套：

**Example:**

***Unacceptable***

  LOOP AT gt\_itab INTO ls\_itab.  
    IF a EQ b.  
      IF c EQ d.  
        IF e EQ f.  
          ......  
        ENDIF.  
      ELSE.  
        ......  
      ENDIF.  
    ELSE.  
      CONTINUE.  
    ENDIF.  
  ENDLOOP.

***Acceptable***

  LOOP AT gt\_itab INTO ls\_itab.  
    CHECK a EQ b.  
  
    IF c EQ d.  
      IF e EQ f.  
        ......  
      ENDIF.  
    ELSE.  
      ......  
    ENDIF.  
  ENDLOOP.

### Loop

LOOP语句的结构和对齐方式如下例：

**Example:**

  LOOP AT i\_tab INTO w\_tab WHERE value LE l\_field  
                 AND flag  EQ space.  
    ...  
  ENDLOOP.

(请注意以上变量和关键字的对齐方式)

### Read

READ语句的结构和对齐方式如下例：

**Example:**

  READ TABLE i\_tab WITH KEY key1 = l\_key1\_value  
                  key2 = l\_key2\_value  
                 TRANSPORTING NO FIELDS  
                 BINARY SEARCH.

(请注意以上变量和关键字的对齐方式)

### While

WHILE语句的结构和对齐方式如下例：

**Example:**

  WHILE l\_index1 LE l\_index2   
    AND l\_index3 GE l\_index4.  
    ...  
  ENDWHILE.

(请注意以上变量和关键字的对齐方式)

### Do

DO语句的结构和对齐方式如下例：

**Example:**

do l\_count times

...

enddo.

### Case

CASE语句的结构和对齐方式如下例：

**Example:**

case wa\_discon-status.

when c\_discstat\_complete.

...

when c\_discstat\_inprogress.

...

when c\_discstat\_start or c\_discstat\_onhold.

...

endcase.

如果IF..ELSE语句中比较的变量相同，那么应该始终使用CASE语句而不是IF语句。在每个WHEN代码区域中，不允许包含复杂的业务逻辑处理，而应该将这些业务逻辑的处理封装到子程序中，通过perform 语句调用。

***Unacceptable***

case wa\_discon-status.

when c\_discstat\_complete.

loop at i\_disconn into w\_disc.

if w\_disc-ab < g\_sysdate.

...

endif.

read table i\_tab with key field = w\_disc-vkont.

...

endloop.

when ...

...

endcase.

***Acceptable***

case wa\_discon-status.

when c\_discstat\_complete.

perForm frm\_check\_disconn\_data.

when ...

...

endcase.

Form frm\_check\_disconn\_data.

loop at i\_disconn into w\_disc.

if w\_disc-ab < g\_sysdate.

...

endif.

read table i\_tab with key field = w\_disc-vkont.

...

endloop.

endform.

### Concatenate

CONCATENATE语句的结构和对齐方式如下例：

**Example:**

  CONCATENATE l\_title       "bp salutation (mr./mrs. etc)  
         l\_firstname   "bp first name  
         l\_lastname    "bp last name  
        INTO l\_output\_field  
     SEPARATED BY l\_separator.

(请注意以上变量和关键字的对齐方式，以及相应的注释)

# 报表程序规范

## 程序结构

所有可执行程序应该具有以下类似结构：

\*&---------------------------------------------------------------------\*  
\*& Report  
\*&---------------------------------------------------------------------\*  
  
\*&---------------------------------------------------------------------\*  
\*& Responsibility  
\*&---------------------------------------------------------------------\*  
\* Program Name:  
\* Date written:  
\* Author's name:  
\* Last update:  
\* Program title:  
\* Project Name:  
\* Version:  
  
\*&---------------------------------------------------------------------\*  
\* Description: (Incl. Related Function Area and System)  
\*&---------------------------------------------------------------------\*  
\*  
  
\*&---------------------------------------------------------------------\*  
\* Change History  
\*&---------------------------------------------------------------------\*  
\*     Date   |   Programmer   |   Corr. #   |   Description  
\*            |                |             |  
\*            |                |             |  
\*&---------------------------------------------------------------------\*  
  
REPORT  MESSAGE-ID xx NO STANDARD PAGE HEADING LINE-SIZE 150.  
TABLES:  
  
\*&---------------------------------------------------------------------\*  
\* TYPES  
\*&---------------------------------------------------------------------\*  
  
\*&---------------------------------------------------------------------\*  
\* CONSTANTS  
\*&---------------------------------------------------------------------\*  
  
\*&---------------------------------------------------------------------\*  
\* GLOBAL DATA  
\*&---------------------------------------------------------------------\*  
  
\*&---------------------------------------------------------------------\*  
\* GLOBAL INTERNAL TABLES  
\*&---------------------------------------------------------------------\*  
  
\*&---------------------------------------------------------------------\*  
\* GLOBAL RANGES  
\*&---------------------------------------------------------------------\*  
  
\*&---------------------------------------------------------------------\*  
\* SELECTION-SCREEN  
\*&---------------------------------------------------------------------\*  
  
\*&---------------------------------------------------------------------\*  
\* GLOBAL MACROS  
\*&---------------------------------------------------------------------\*  
  
\*&---------------------------------------------------------------------\*  
\* EVENTS BEFORE MAIN PROGRAM  
\*&---------------------------------------------------------------------\*  
\*INITIALIZATION.  
  
\*AT SELECTION-SCREEN OUTPUT.  
  
\*AT SELECTION-SCREEN ON VALUE-REQUEST FOR.  
  
\*AT SELECTION-SCREEN.  
  
\*TOP-OF-PAGE.  
  
\*&---------------------------------------------------------------------\*  
\*  
\*        MAIN PROGRAM  
\*  
\*----------------------------------------------------------------------\*  
START-OF-SELECTION.  
\* Initialize data  
  PERForm frmrm\_initialize\_data.  
  
\* Process input data (for inbound interfaces)  
  PERForm frmrm\_process\_input\_data.  
  
\* Select data from database  
  PERForm frmrm\_select\_data.  
  
\* Process selected data  
  PERForm frmrm\_process\_data.  
  
END-OF-SELECTION.  
\* Combine results from the processing (may not be necessary)  
  PERForm frmrm\_combine\_data.  
  
\* Output result to file or spool/screen  
  PERForm frmrm\_output\_data.

程序框架应该符合以下的层次结构，也就是说每个子程序应该把其中的逻辑打散到更小的子程序中，以使子程序的逻辑简单化，增加代码的可重用性和可靠性。

**Example:**

典型的程序层次结构如下：

|\_\_perForm frmrm\_select\_data

| |

| |\_\_ perForm frmrm\_select\_device\_data.

| | |

| | |\_\_ perForm frmrm\_select\_equi

| | |

| | |\_\_ perForm frmrm\_select\_egerh

| | |

| | |\_\_ perForm frmrm\_select\_egers

| |

| |\_\_ perForm frmrm\_select\_inst\_data.

|

|\_\_ perForm frmrm\_process\_data

| |

| |\_\_ perForm frmrm\_process\_mru\_data.

| |

| |\_\_ perForm frmrm\_process\_device\_data.

|

|\_\_ perForm frm\_output\_data.

### 数据初始化子程序

数据初始化子程序主要用于数据读取、处理所需的初始化操作，比如初始化range变量、系统变量、全局变量等。如果需要对多个range变量进行初始化，那么需要将它们分别放到各自的子程序中处理。

**Example:**

Form frmrm\_initialize\_data.

\* Initialize system fields

perForm frmrm\_init\_data\_sysfields.

\* Initialize ranges

perForm frmrm\_init\_data\_ranges.

endForm frmrm\_initialize\_data.

Form frmrm\_init\_data\_sysfields.

g\_sysdate = sy-datum.

g\_sysuname = sy-uname.

endform

Form frmrm\_init\_data\_ranges.

r\_divisions = c\_range\_search\_eq. “c\_range\_search\_eq = ‘IEQ’

r\_division-low = c\_div\_elec.

append r\_division.

endform

### 数据存取子程序

此子程序作为数据存取的唯一入口，通过perform来调用各个select子程序。所有的数据存取语句（select）都应该包含在相应的select子程序中；通常情况下，一个select语句应该包含在一个独立的子程序中。

**Example:**

Form frmrm\_select\_data.

\* Get MRUs

perForm frmrm\_select\_mru.

\* Get Installations.

perForm frmrm\_select\_inst.

Endform.

Form frmrm\_select\_mru.

\* Get all MRUs

Select ableinh

Into i\_mrus

From te422

Where adatsoll in s\_adatsoll.

endform.

Form frmrm\_select\_inst.

Check i\_mrus[] is not initial.

\* Get all installations

Select anlage

Into i\_insts

From eanlh

For all entries in i\_mrus

Where ableinh = i\_mrus-anlage.

endform.

### 数据处理子程序

该子程序应该是处理数据的唯一入口，所有逻辑处理的子程序应该通过该子程序来调用。

**Example:**

Form frmrm\_process\_data.

\* Process MRUs

perForm frmrm\_process\_mru.

\* Process Installation Data

perForm frmrm\_process\_inst.

Endform.

Form frmrm\_process\_mru.

\* Process MRUs that start with ‘M\*’

perForm frm\_process\_mru\_main.

\* Process All other MRUs that don’t start with ‘M\*’

perForm frmrm\_process\_mrus\_sub.

endform.

Form frmrm\_process\_inst.

...

endform.

### 数据组合子程序

该子程序仅用于将F\_PROCESS\*处理的结果组合成最终的输出结构。

该子程序不是必需的。

**Example:**

Form frmrm\_combine\_data.

Loop at i\_inst into w\_inst.

Read table i\_easts with key anlage = w\_inst-anlage

Binary search.

....

w\_output-anlage = w\_inst-anlage.

w\_output-device = w\_easts-device.

Append w\_output to i\_output.

Endloop.

Endform.

### 数据输出子程序

该子程序仅用于输出结果。所有相关的数据处理逻辑不应该出现在该子程序中，而应该出现在F\_PROCESS\* 子程序中。该子程序的作用只是用于格式化输出。

**Example:**

Form frmrm\_output\_data.

\* Output data

perForm frmrm\_write\_column\_heading.

format color col\_red.

Loop at i\_output into w\_output.

write at: /g\_col1 w\_output-anlage “installation

g\_col2 w\_output-device. “device

Endloop.

form color off.

Endform.

# 子程序规范

子程序是代码模块化的非常好的一种方法。

## 模块化代码

使用子程序的主要目的是使程序代码模块化。程序中使用的公共代码应该转换为子程序，而不是在每个地方使用代码拷贝。

**Example:**

*Assume the following declarations*

types: begin of ty\_devmr,

equnr like eabl-equnr, “Equipment number

adatsoll like eabl-adatsoll, “Schedule MR Date

flag type c, “Indicate old MR or not

end of ty\_devmr.

types: ty\_tab\_devmr type ty\_devmr occurs 0.

data: i\_devmr1 type ty\_tabmr.

data: i\_devmr2 type ty\_tabmr.

***Unacceptable*:**

Form frmrm\_check\_device\_lists

...

loop at **i\_devmr1** assigning <fs\_mr> where adatsoll < g\_sysdate.

<fs\_mr>-flag = c\_true.

endloop.

loop at **i\_devmr2** assigning <fs\_mr> where adatsoll < g\_sysdate.

<fs\_mr>-flag = c\_true.

endloop.

...

Endform.

***Acceptable*:**

Form frmrm\_check\_device\_lists

...

perForm frm**rm\_check\_single\_dev\_list** tables i\_devmr1.

perForm frm**rm\_check\_single\_dev\_list** tables i\_devmr2.

...

Endform.

Form frm**rm\_check\_single\_dev\_list** tables pi\_devlist type ty\_tab\_devmr.

\* Loop at the list of input devices and set the flag.

loop at pi\_devlist assigning <fs\_mr> where adatsoll < g\_sysdate.

<fs\_mr>-flag = c\_true.

endloop.

Endform.

## 参数

### 参数命名

子程序参数命名参考2.2.2变量命名规范。

### 参数类型

所有传入子程序的参数都必须指定参数类型。

***Unacceptable*:**

Form frmrm\_process\_billdoc using pwa\_billdoc.

...

Endform.

***Acceptable*:**

Form frmrm\_process\_billdoc using pwa\_billdoc **type ty\_billdoc.**

...

Endform.

### Table参数

通常情况下，数据量大的内表应该定义为全局变量而不应该通过参数来传递。而对于数据量小的内表，有时候则是通过子程序参数来传递（程序员应该在这一点上合理把握）。参数类型通常情况下使用**type**而不是**structure**来定义。

***Unacceptable*:**

Form frmrm\_process\_billdoc tables pi\_docs structure i\_docs.

...

Endform.

***Acceptable*:**

Form frmrm\_process\_billdoc tables pi\_docs **type ty\_tab\_doclist.**

...

Endform.

### 参数个数要求

当有大量的有关联的简单变量需要传入时，最好是采用以下方式之一：

* 创建一个公用的结构，以结构的方式传入参数
* 通过小的内表来传入值
* 使用range变量来传入值

程序员应该合理判断采用哪种方式更合适。

***Unacceptable*:**

...

perForm frmrm\_process\_date using l\_date1

l\_date2

l\_datecheck\_a

l\_date3

l\_date4

l\_datecheck\_b.

...

Form frmrm\_process\_date using l\_startdate\_a like sy-datum

l\_enddate\_a like sy-datum

l\_datecheck\_a type c

l\_startdate\_b like sy-datum

l\_enddate\_b like sy-datum

l\_datecheck\_b type c.

...

if l\_startdate\_a < sy-datum

and l\_enddate\_a > sy-datum

and l\_datecheck\_a = c\_true

and l\_startdate\_b < sy-datum

and l\_enddate\_b > sy-datum

and l\_datecheck\_b = c\_true.

...

Endif.

...

Endform.

***Acceptable*:**

...

types: begin of ty\_date,

start like sy-datum, “Start Date

end like sy-dateum, “End Date

check type c, “Date check flag

end of ty\_date.

...

\* lwa\_date1 and lwa\_date2 are of type ty\_date and contain the

\* the required dates.

perForm frm\_process\_date using lwa\_date1

　 lwa\_date2.

...

Form frm\_process\_date using pwa\_date1 type ty\_date

pwa\_date2 type ty\_date.

if pwa\_date1-start < sy-datum

and pwa\_date1-end > sy-datum

and pwa\_date1-check = c\_true

and pwa\_date2-start < sy-datum

and pwa\_date2-end > sy-datum

and pwa\_date2-check = c-true.

...

Endif.

Endform.

### 参数对齐

子程序使用的形式变量必须按照以下格式对齐。

***Unacceptable*:**

Form frmrm\_process\_date tables pi\_tab type ty\_tab\_tab

using l\_date like sy-datum

l\_datecheck type c.

...

Endform.

***Acceptable*:**

Form frmrm\_process\_date tables pi\_tab type ty\_tab\_tab

using l\_date like sy-datum

　 l\_datecheck type c.

...

Endform.

## 子程序结构

每一个子程序应该只包含一个逻辑功能块，如果需要多个逻辑块，那么该子程序应该拆分为多个子程序。

* 每一个select语句应该有一个单独的子程序（关系非常紧密的多个select语句可以放到一个子程序中）
* 如果子程序中包含两个以上的连续loop语句，那么应该把它们拆分到不同的子程序中
* 如果一个子程序中包含的代码行超过100，那么应该把该子程序拆分成多个子程序

***Unacceptable*:**

Form frm\_process\_data.

Loop at i\_movein into w\_movein.

...

Endloop.

Loop at i\_inst into w\_inst.

...

Endloop.

Endform.

***Acceptable*:**

Form frm\_process\_data.

\* Process the move-in records

PerForm frm\_process\_movein.

\* Process the installation records

PerForm frm\_process\_insts.

Endform.

Form frm\_process\_movein.

Loop at i\_movein into w\_movein.

...

Endloop.

Endform.

Form frm\_process\_inst.

Loop at i\_inst into w\_inst.

...

Endloop.

Endform.

## Perform语句

使用perform语句调用子程序时，传递的参数必须对齐且注释；每一个参数应该分行列出。

***Unacceptable*:**

perForm frm\_delete\_unused\_bill\_orders tables i\_billorders using l\_date.

***Acceptable*:**

\* Use the standard function to delete unused billing orders from the list

\* in I\_BILLORDERS

perForm frm\_delete\_unused\_bill\_orders tables i\_billorders “Bill order list

using l\_date. “Input date

## 子程序注释

所有的子程序必须包含注释以说明子程序的功能以及输入输出参数的含义。

**Example:**

\*&---------------------------------------------------------------------\*

\*& Form F\_CONV\_DATE

\*----------------------------------------------------------------------\*

\* This is used to convert the date in the yyyymmdd format into the \*

\* the text string: dd.mm.yyyy format. \*

\*----------------------------------------------------------------------\*

\* --> p\_inp\_date = The original date in the yyyymmdd format \*

\* <-- p\_out\_date = The date in the dd.mm.yyyy format. \*

\*----------------------------------------------------------------------\*

Form frm\_conv\_dat using p\_inp\_date like sy-datum

　 p\_out\_date(10) type c.

Endform.

# 功能模块规范

## 函数模块结构

函数模块的主要代码区域（FUNCTION和ENDFUNCTION之间) 不能包含复杂的逻辑：

* 只能包含子程序或其他函数的调用语句
* 可以包含简单的 IF, CASE, LOOP 等语句
* 所有的复杂的逻辑处理（包含select）必须定义为子程序中，放入相应的函数模块包含程序中
* 如果函数模块中全部的代码（包含注释文本）不超过100行，那么可以例外

## 函数返回值

函数调用后必须检查相应的返回状态变量SY-SUBRC，如果确实不需要检查，应该说明原因。

## 函数调用

调用函数模块时，如果有不使用的IMPORTING/EXPORTING/TABLE参数，请删除相应的注释行。

同时，所有的参数必须格式对齐。

***Unacceptable*:**

CALL FUNCTION 'BAPI\_IDENTIFICATIONDETAILS\_GET'

EXPORTING

BUSINESSPARTNER = W\_EBILL\_CA-GPART

TABLES

\* IDENTIFICATIONDETAIL = I\_BAPIBUS1006\_ID\_DETAILS

RETURN = I\_BAPIRET2.

***Acceptable*:**

CALL FUNCTION 'BAPI\_IDENTIFICATIONDETAILS\_GET'

EXPORTING

BUSINESSPARTNER = W\_EBILL\_CA-GPART

TABLES

RETURN = I\_BAPIRET2.

*(Please note, that all the fields are aligned neatly)*

# 屏幕对话程序规范

## 屏幕逻辑流结构

所有自定义的主屏幕逻辑流均需定义为以下类似结构；子屏幕不应包含：TOOLBAR、　USER-COMMAND和EXIT-COMMAND几个部分

PROCESS BEFORE OUTPUT.

\* Set toolbar and titlebar

MODULE status\_xxxx. “xxxx-screen number

\* Set output attribute for screen fields

MODULE dynpro\_fields\_xxxx. “xxxx-screen number

\* move data from program to screen fields

MODULE dynpro\_data\_out\_xxxx. “xxxx-screen number

\* Table control loop

LOOP AT gt\_tab INTO gs\_tab WITH CONTROL tc\_xxxx CURSOR tc\_xxxx-current\_line.

\* Set output attribute of table control fields

MODULE dynpro\_fields\_tc\_xxxx.

\* move data from program to table control fields

MODULE dynpro\_data\_out\_tc\_xxxx.

ENDLOOP.

PROCESS AFTER INPUT.

\* Process EXIT-COMMAND

MODULE exit\_command\_xxxx AT EXIT-COMMAND. “xxxx-screen number

\* check input validation of screen fields

CHAIN.

FIELD: a,

b.

MODULE dynpro\_data\_check\_<w>\_xxxx ON CHAIN-REQUEST. “<w>-meaning words

ENDCHAIN.

\* move data from screen fields to program

CHAIN.

FIELD: a,

b,

c.

MODULE dynpro\_data\_in\_xxxx ON CHAIN-REQUEST. “xxxx-screen number

ENDCHAIN.

\* Table control loop

LOOP AT gt\_tab.

\* check input validation of table control fields

CHAIN.

FIELD: c,

d.

MODULE dynpro\_data\_check\_<w>\_tc\_xxxx ON CHAIN-REQUEST.“<w>-meaning words

ENDCHAIN.

\* move data from table control fields to program

CHAIN.

FIELD: c,

d,

e.

MODULE dynpro\_data\_in\_tc\_xxxx ON CHAIN-REQUEST.

ENDCHAIN.

ENDLOOP.

\* Process USER-COMMAND

MODULE user\_command\_xxxx. “xxxx-screen number

PROCESS ON VALUE-REQUEST.

\* Process f4 for screen field a

FIELD a MODULE dynpro\_f4\_a\_xxxx. “xxxx-screen number

## PAI/PBO/POV 模块

所有屏幕模块必须包含以下类似结构

MODULE <module\_name> [INPUT|OUTPUT].

\* Call subroutine of each special module

PERForm frm\_<module\_name>\_[pai|pbo|pov].

ENDMODULE.

在〔MODULE…..ENDMODULE〕中不允许有任何逻辑处理语句，更不能定义任何类型的变量，只允许使用PERFORM语句调用特定的子程序。所有相关的业务逻辑只能包含在该子程序或其调用的下层子程序中.

**Example:**

***Unacceptable*:**

MODULE dynpro\_data\_out\_0100 OUTPUT.

ekko-ebeln = gs\_ekko-ebeln.

ekko-lifnr = gs\_ekko-lifnr.

SELECT SINGLE name1 INTO dynpro\_name1 FROM lfa1 WHERE lifnr = ekko-lifnr.

ENDMODULE.

***Acceptable*:**

MODULE dynpro\_data\_out\_0100 OUTPUT.

PERForm frm\_dynpro\_data\_out\_0100\_pbo.

ENDMODULE.

Form frm\_dynpro\_data\_out\_0100\_pbo.

ekko-ebeln = gs\_ekko-ebeln.

ekko-lifnr = gs\_ekko-lifnr.

SELECT SINGLE name1 INTO dynpro\_name1 FROM lfa1 WHERE lifnr = ekko-lifnr.

ENDFORM.

### User-command模块

这个模块只能用于处理除EXIT-COMMAND以外的所有用户按钮操作；在该模块调用的子程序中，只允许使用CASE判断屏幕触发的按钮事件，任何业务逻辑语句均需要包含在相应的子程序中。相应子程序应按照以下规则命名：

Frm\_fcode\_<okcode>\_<dynnr>

<okcode>－按钮名称

<dynnr>－屏幕号

所有屏幕事件代码均使用全局变量ok\_code接收，同时必须定义全局变量ok\_save，在user\_command模块中按照以下方式来处理

**EXAMPLE:**

Form frm\_user\_command\_0100\_pai.

OK\_SAVE = OK\_CODE.

CLEAR OK\_CODE.

CASE OK\_SAVE.

WHEN ‘ADD’. “add data line

perForm frm\_fcode\_add\_0100.

WHEN ‘SAVE’. “save data

perForm frm\_fcode\_save\_0100.

WHEN OTHERS.

ENDCASE.

Endform.

# BADI增强规范

实施BADI增强需按照以下步骤实现：

|  |  |
| --- | --- |
| **Step** | **Description** |
| **1** | 在SE19中创建客BADI实施：命名参考本文档“1.5 BADI实施命名规范” |
| **2** | 在相应的BADI实施类中，增加自定义方法，所有业务逻辑的处理均应包含在自定义方法中：Z<XX>\_<BADI\_METH>\_<nn>; |
| **3** | 在BADI实施标准方法中使用call method调用相应的自定义业务处理方法 |

注意：在BADI实施模块的标准方法中，不能有复杂的业务逻辑

**Example:**

***Unacceptable*:**

METHOD if\_ex\_ac\_document~change\_initial.  
  
     LOOP AT im\_document-item INTO w\_item.  
  
      w\_ebeln = w\_item-ebeln.  
      w\_ebelp = w\_item-ebelp.  
  
      CHECK w\_ebeln IS NOT INITIAL AND w\_ebelp IS NOT INITIAL.  
  
  
      SELECT SINGLE \* FROM ekpo

WHERE ebeln EQ w\_ebeln

AND ebelp EQ w\_ebelp.

......

    ENDLOOP.  
  
  
ENDMETHOD.                    "if\_ex\_ac\_document~change\_initial

***Acceptable*:**

METHOD if\_ex\_ac\_document~change\_initial.

\* Change document data

CALL METHOD ZSG1\_CHANGE\_INITIAL\_01

EXPORTING

IM\_DOCUMENT = IM\_DOCUMENT

IMPORTING

EX\_DOCUMENT = EX\_DOCUMENT.

ENDMETHOD.                    "if\_ex\_ac\_document~change\_initial

METHOD ZSG1\_CHANGE\_INITIAL\_01

\* Get data from purchasing order

CALL METHOD ZFILL\_DOC\_ITEM\_WITH\_PO\_ITEM.

EXPORTING

IM\_DOCUMENT = IM\_DOCUMENT

IMPORTING

EX\_DOCUMENT = EX\_DOCUMENT.

\* Other enhancement data

CALL METHOD ZFILL\_DOC\_ITEM\_WITH\_CUSTOM\_DATA

EXPORTING

IM\_DOCUMENT = IM\_DOCUMENT

IMPORTING

EX\_DOCUMENT = EX\_DOCUMENT.

ENDMETHOD.                    "ZSG1\_CHANGE\_INITIAL\_01

METHOD ZFILL\_DOC\_ITEM\_WITH\_PO\_ITEM

\* Get data from purchasing order

LOOP AT im\_document-item INTO w\_item.  
  
      w\_ebeln = w\_item-ebeln.  
      w\_ebelp = w\_item-ebelp.  
  
      CHECK w\_ebeln IS NOT INITIAL AND w\_ebelp IS NOT INITIAL.  
  
  
      SELECT SINGLE \* FROM ekpo

WHERE ebeln EQ w\_ebeln

AND ebelp EQ w\_ebelp.

......

  ENDLOOP.

ENDMETHOD.                    "ZFILL\_DOC\_ITEM\_WITH\_PO\_ITEM.

# CMOD增强规范

CMOD中的增强用户出口包含一个函数模块 (EXIT\_\*) 和一个可以客户化修改的include程序。**该include程序中不允许包含任何逻辑**！

为了实施一个增强，应该将用户出口函数模块 (EXIT\_\*) 拷贝为一个用户函数模块（按命名规范命名z\*），所有的逻辑处理应该包含在这个拷贝出的函数模块中；而出口inlucde程序，仅需要调用该函数的语句。

例如，假定有一个用户出口 EXIT\_SAPLES55\_001，它包含一个inlcude程序ZXSAPL02；那么应该按照以下步骤来实施这个增强出口：

|  |  |
| --- | --- |
| **Step** | **Description** |
| **1** | 在CMOD中创建一个项目，包含所需的增强 |
| **2** | 拷贝用户出口函数模块EXIT\_SAPLES55\_001为一个用户函数模块 (例如，新的函数模块名为ZISCS\_EXITES55\_COBJ) |
| **3** | 在函数模块ZISCS\_EXITES55\_COBJ中加入所需的增强逻辑 |
| **4** | 在inlude文件ZXSAPL02中调用函数模块ZISCS\_EXITES55\_COBJ，并传递函数模块EXIT\_SAPLES55\_001的所有参数 |

如果需要在用户出口中增加一块全新的逻辑，那么应该拷贝出新的函数模块来实现。

# BTE增强规范

所有BTE的开发均必须在客户定制范围内实现，不允许对SAP标准应用的BTE做任何更改；实施BTE增强需按照以下步骤实现：

|  |  |
| --- | --- |
| **Step** | **Description** |
| **1** | 在FIBF中创建客户定制产品PROD：Z<XX>\_<nnn>，如：ZFIN\_001 |
| **2** | 拷贝相应BTE函数模版到自定义函数模块：<PROD>\_<event>\_<desc>，如：ZFIN\_001\_1030\_FIDOC\_SAVE |
| **3** | 在函数模块ZFIN\_001\_1030\_FIDOC\_SAVE中增加相应的逻辑，函数中的代码必须符合本文档中“功能模块规范“要求 |
| **4** | 在FIBF相应P/S Module或Process Module中客户产品部分指定：事件、产品、函数模块，如：00001030 | ZFIN\_001 | ZFIN\_001\_1030\_FIDOC\_SAVE |

# 程序性能相关代码规范

## SQL语句

* 始终把条件放在where子句中而不要使用check语句来检查，这样数据库系统可以使用索引（如果可能），同时这样可以大大降低网络负载。

**Example:**

***Unacceptable:***

SELECT \* FROM SBOOK INTO SBOOK\_WA.

CHECK: SBOOK\_WA-CARRID = 'LH' AND

SBOOK\_WA-CONNID = '0400'.

ENDSELECT.

***acceptable:***

SELECT \* FROM SBOOK INTO SBOOK\_WA

WHERE CARRID = 'LH' AND

CONNID = '0400'.

ENDSELECT.

* 如果需要检查表或视图是否有满足某条件的记录存在，请使用Select ... Up To 1 Rows 语句，而不要使用带Exit的Select-Endselect-loop语句。如果条件中包含所有的主键字段，则应该使用Select Single语句。Select Single语句仅需要与数据库通信一次，而 Select-Endselect则需要两次。

**Example:**

***Unacceptable:***

SELECT \* FROM SBOOK INTO SBOOK\_WA

WHERE CARRID = 'LH'.

EXIT.

ENDSELECT.

***acceptable:***

SELECT \* FROM SBOOK INTO SBOOK\_WA

UP TO 1 ROWS

WHERE CARRID = 'LH'.

ENDSELECT.

* 如果想查询最大值、最小值、总和、平均值或者条目数，则应该使用统计函数；这样可以大大降低网络负载。

**Example:**

***Unacceptable:***

DATA: MAX\_MSGNR type t100-msgnr.

MAX\_MSGNR = '000'.

SELECT \* FROM T100 INTO T100\_WA

WHERE SPRSL = 'D' AND

ARBGB = '00'.

CHECK: T100\_WA-MSGNR > MAX\_MSGNR.

MAX\_MSGNR = T100\_WA-MSGNR.

ENDSELECT.

***acceptable:***

DATA: MAX\_MSGNR type t100-msgnr.

SELECT MAX( MSGNR ) FROM T100 INTO max\_msgnr

WHERE SPRSL = 'D' AND

ARBGB = '00'.

* 如果仅需读取表或视图中的个别字段，那么应该使用select列表而不要使用Select \* ；这样可以大大降低网络负载。

**Example:**

***Unacceptable:***

SELECT \* FROM DD01L INTO DD01L\_WA

WHERE DOMNAME LIKE 'CHAR%'

AND AS4LOCAL = 'A'.

ENDSELECT.

***acceptable:***

SELECT DOMNAME FROM DD01L

INTO DD01L\_WA-DOMNAME

WHERE DOMNAME LIKE 'CHAR%'

AND AS4LOCAL = 'A'.

ENDSELECT.

* 更新数据库时，尽可能使用列更新而不要使用单行更新。这样可以大大降低网络负载。

**Example:**

***Unacceptable:***

SELECT \* FROM SFLIGHT INTO SFLIGHT\_WA.

SFLIGHT\_WA-SEATSOCC =

SFLIGHT\_WA-SEATSOCC - 1.

UPDATE SFLIGHT FROM SFLIGHT\_WA.

ENDSELECT.

***acceptable:***

UPDATE SFLIGHT

SET SEATSOCC = SEATSOCC - 1.

* 对于经常使用的select语句，尽量使用索引。在select语句的where子句中，尽量使用通过AND连接的索引字段，这样数据库总是会使用索引。注意：过于复杂的where子句对于数据库系统语句优化器来说都是不利于处理的。

**Example:**

***Unacceptable:***

SELECT \* FROM SBOOK CLIENT SPECIFIED INTO SBOOK\_WA

WHERE CARRID = 'LH'

AND CONNID = '0400'.

ENDSELECT.

***acceptable:***

SELECT \* FROM SBOOK CLIENT SPECIFIED INTO SBOOK\_WA

WHERE MANDT IN ( SELECT MANDT FROM T000 )

AND CARRID = 'LH'

AND CONNID = '0400'.

ENDSELECT.

* 对于经常使用的只读表，尽量使用SAP缓存。可以大大降低网络负载。

**Example:**

***Unacceptable:***

SELECT SINGLE \* FROM T100 INTO T100\_WA

BYPASSING BUFFER

WHERE SPRSL = 'D'

AND ARBGB = '00'

AND MSGNR = '999'.

***acceptable:***

SELECT SINGLE \* FROM T100 INTO T100\_WA

WHERE SPRSL = 'D'

AND ARBGB = '00'

AND MSGNR = '999'.

* 一般情况下使用Into Table总是会快于Select循环中使用Append 语句

**Example:**

***Unacceptable:***

DATA T006\_WA TYPE T006.

CLEAR X006.

SELECT \* FROM T006 INTO T006\_WA.

APPEND T006\_WA TO X006.

ENDSELECT.

***acceptable:***

SELECT \* FROM T006 INTO TABLE X006.

* 更新数据库表时，尽可能使用数组操作而不要使用单行操作。应用程序和数据库系统频繁通信会造成大量的系统负载。

**Example:**

***Unacceptable:***

LOOP AT TAB INTO TAB\_WA.

INSERT INTO CUSTOMERS VALUES TAB\_WA.

ENDLOOP.

***acceptable:***

INSERT CUSTOMERS FROM TABLE TAB.

* 如果选择的数据只处理一次，那么应该使用Select-Endselect循环，二不要使用Into Table把数据收集到内表中再处理；内表处理会占用更多的空间。

**Example:**

***Unacceptable:***

SELECT \* FROM T006

INTO TABLE X006.

LOOP AT X006 INTO X006\_WA.

ENDLOOP.

***acceptable:***

SELECT \* FROM T006 INTO X006\_WA.

ENDSELECT.

* 如果要处理表连接，应该使用视图而不要使用嵌套的select语句；这样可以有效的降低网络负载。

**Example:**

***Unacceptable:***

SELECT \* FROM DD01L INTO DD01L\_WA

WHERE DOMNAME LIKE 'CHAR%'

AND AS4LOCAL = 'A'.

SELECT SINGLE \* FROM DD01T INTO DD01T\_WA

WHERE DOMNAME = DD01L\_WA-DOMNAME

AND AS4LOCAL = 'A'

AND AS4VERS = DD01L\_WA-AS4VERS

AND DDLANGUAGE = SY-LANGU.

ENDSELECT.

***acceptable:***

SELECT \* FROM DD01V INTO DD01V\_WA

WHERE DOMNAME LIKE 'CHAR%'

AND DDLANGUAGE = SY-LANGU.

ENDSELECT.

* 如果要从几个逻辑关联的表中读取数据，应该使用表连接而不要使用嵌套的select语句；这样可以有效的降低网络负载。

**Example:**

***Unacceptable:***

SELECT \* FROM SPFLI INTO SPFLI\_WA.

SELECT \* FROM SFLIGHT INTO SFLIGHT\_WA

WHERE CARRID = SPFLI\_WA-CARRID

AND CONNID = SPFLI\_WA-CONNID.

ENDSELECT.

ENDSELECT.

***acceptable:***

SELECT \* INTO WA

FROM SPFLI AS P INNER JOIN SFLIGHT AS F

ON P~CARRID = F~CARRID AND

P~CONNID = F~CONNID.

ENDSELECT.

* 通常情况下嵌套的Select 循环或者FOR ALL ENTRIES 都可以通过子查询来实现；子查询可以有效的降低网络负载。

**Example:**

***Unacceptable:***

SELECT \* FROM SPFLI

INTO TABLE T\_SPFLI

WHERE CITYFROM = 'FRANKFURT'

AND CITYTO = 'NEW YORK'.

SELECT \* FROM SFLIGHT AS F

INTO SFLIGHT\_WA

FOR ALL ENTRIES IN T\_SPFLI

WHERE SEATSOCC < F~SEATSMAX

AND CARRID = T\_SPFLI-CARRID

AND CONNID = T\_SPFLI-CONNID

AND FLDATE BETWEEN '19990101' AND '19990331'.

ENDSELECT.

***acceptable:***

SELECT \* FROM SFLIGHT AS F INTO SFLIGHT\_WA

WHERE SEATSOCC < F~SEATSMAX

AND EXISTS ( SELECT \* FROM SPFLI

WHERE CARRID = F~CARRID

AND CONNID = F~CONNID

AND CITYFROM = 'FRANKFURT'

AND CITYTO = 'NEW YORK' )

AND FLDATE BETWEEN '19990101' AND '19990331'.

ENDSELECT.

## 内表操作

* 如果内表的记录条数很多（>20），那么对全表进行线性查询是非常费时的。应该尽量使内表排序，并使用二分法搜索，或者使用SORTED TABLE。假如内表含有n条记录，线性搜索执行时间为O( n ) ，而二分法执行时间仅为O( log2( n ) )。

**Example:**

***Unacceptable:***

\* Entries: 1000, Line width: 100

\* Key width: 20

\* The READ ends with SY-SUBRC=4

READ TABLE ITAB INTO WA

WITH KEY K = 'X'.

***acceptable:***

\* Entries: 1000, Line width: 100

\* Key width: 20

\* The READ ends with SY-SUBRC=4

READ TABLE ITAB INTO WA

WITH KEY K = 'X'

BINARY SEARCH.

* 如果需要使用不同的关键字重复读取内表，那么应该在代码中保存相应的次级索引。有了次级索引就可以使用二分法搜索加上一个索引读取来替代线性搜索。

**Example:**

***Unacceptable:***

\* Entries: 1000, Line width: 100

\* Key width: 20

\* The READ locates the 500th entry.

READ TABLE ITAB INTO WA

WITH KEY DATE = SY-DATUM.

IF SY-SUBRC = 0.

" ...

ENDIF.

***acceptable:***

\* Entries: 1000, Line width: 100

\* Key width: 20

\* The READ locates the 500th entry.

READ TABLE SEC\_IDX INTO SEC\_IDX\_WA

WITH KEY DATE = SY-DATUM

BINARY SEARCH.

IF SY-SUBRC = 0.

READ TABLE ITAB INTO WA

INDEX SEC\_IDX\_WA-INDX.

" ...

ENDIF.

* LOOP ... WHERE比LOOP/CHECK性能更好，因为LOOP ... WHERE由内部来判断指定的条件。同所有的逻辑表达式一样，如果操作数的类型相同，则性能会更好。可能的情况下，使用FROM i1且（或者）TO i2可以进一步提高性能。

**Example:**

***Unacceptable:***

\* Entries: 1000, Line width: 500

\* Key width: 20

\* 5 entries of which match the key condition

LOOP AT ITAB INTO WA.

CHECK WA-K = 'X'.

" ...

ENDLOOP.

***acceptable:***

\* Entries: 1000, Line width: 500

\* Key width: 20

\* 5 entries of which match the key condition

LOOP AT ITAB INTO WA WHERE K = 'X'.

" ...

ENDLOOP.

* 排序表中的记录通过二分法来定位，执行的时间依赖于表中记录数（O (log n)）。而哈希表中记录通过哈希算法来定位，执行的时间始终是固定的（O (1)）, 而与表的大小无关。哈希表适用于单条记录的读取，而排序表则适用于符合部分条件的顺序的表循环操作。

**Example:**

***Unacceptable:***

\* Entries: 1000

\* Line width: 100, key width: 20

\* STAB is a unique sorted table, 250 entries are read

DO 250 TIMES.

N = 4 \* SY-INDEX.

READ TABLE STAB INTO WA WITH TABLE KEY K = N.

IF SY-SUBRC = 0.

" ...

ENDIF.

ENDDO.

***acceptable:***

\* Entries: 1000

\* Line width: 100, key width: 20

\* HTAB is a hased table, 250 entries are read

DO 250 TIMES.

N = 4 \* SY-INDEX.

READ TABLE HTAB INTO WA WITH TABLE KEY K = N.

IF SY-SUBRC = 0.

" ...

ENDIF.

ENDDO.

* 哈希表为单条记录读取而优化，记录之间没有特定的顺序关系。因此对哈希表进行部分键值的顺序读取，其性能是没有优化的，而是每一条记录都要检查是否符合条件（即全表搜索）。排序表则是按照主键进行升序排列的。如果查询条件是这样的格式"k1 = v1 AND ... AND kn = vn"（k1 .. kn与表主键的左边部分匹配），那么表的读取是被核心代码优化的，因此只会访问符合条件的记录，而不是全表搜索。

**Example:**

***Unacceptable:***

\* Entries: 10000, Line width: 100

\* key width: 60, Subkey width: 20

\* HTAB is a hashed table, 2000 entries are read

\* Key fields: K, DATA

LOOP AT HTAB INTO WA WHERE K = SUBKEY.

" ...

ENDLOOP.

***acceptable:***

\* Entries: 10000, Line width: 100

\* key width: 60, Subkey width: 20

\* STAB is a sorted table, 2000 entries are read

\* Key fields: K, DATA

LOOP AT STAB INTO WA WHERE K = SUBKEY.

" ...

ENDLOOP.

* 使用MODIFY语句更新内表时，使用" TRANSPORTING f1 f2 ..."可以提高执行的效率；特别是表行越长越明显。

**Example:**

***Unacceptable:***

\* Line width: 500

\* The complete line is moved.

WA-DATE = SY-DATUM.

MODIFY ITAB FROM WA INDEX 1.

***acceptable:***

\* Line width: 500

\* Only the 8 bytes of the selected

\* component are moved.

WA-DATE = SY-DATUM.

MODIFY ITAB FROM WA INDEX 1 TRANSPORTING DATE.

* 使用"LOOP ... ASSIGNING ..."直接存取内表行，可以大大提高内表记录的更新速度。

**Example:**

***Not preferred:***

\* Entries: 100 (outer table), 20 (inner table)

\* The entries to be modified: 50

\* Actually, only the component FLAG is updated.

\* However, the complete lines are moved.

LOOP AT ITAB INTO WA.

I = SY-TABIX MOD 2.

IF I = 0.

WA-FLAG = 'X'.

MODIFY ITAB FROM WA.

ENDIF.

ENDLOOP.

***Preferred:***

\* Entries: 100 (outer table), 20 (inner table)

\* Entries to be modified: 50

\* The component FLAG is updated directly.

LOOP AT ITAB ASSIGNING <WA>.

I = SY-TABIX MOD 2.

IF I = 0.

<WA>-FLAG = 'X'.

ENDIF.

ENDLOOP.

* 使用从下到上的策略来填充内表，那么系统开销会根据数据结构的层次成倍的增加。因为每一个下层内表的数据都需要移动到它们的上层数据结构中。相反，使用从上到下的策略填充内表，先填充外层内表，然后使用"LOOP ... ASSIGNING"直接填充内层内表，这样内层内表的数据只需要移动一次就可以了。

**Example:**

***Not preferred:***

\* Entries: 50 (outer table), 10 (inner tables)

\* Line width: 500 (outer), 4 (inner)

DO 50 TIMES.

CLEAR WA.

DO 10 TIMES.

APPEND N TO WA-INTTAB.

ADD 1 TO N.

ENDDO.

APPEND WA TO ITAB.

ENDDO.

***Preferred:***

\* Entries: 50 (outer table), 10 (inner tables)

\* Line width: 500 (outer), 4 (inner)

DO 50 TIMES.

APPEND INITIAL LINE TO ITAB.

ENDDO.

LOOP AT ITAB ASSIGNING <F>.

DO 10 TIMES.

APPEND N TO <F>-INTTAB.

ADD 1 TO N.

ENDDO.

ENDLOOP.

* 如果需要汇总表记录，那么请使用COLLECT语句 !

**Example:**

***Unacceptable:***

\* TAB1: 1000 entries, Line width: 32

\* 500 entries have different keys

LOOP AT ITAB1 INTO WA1.

READ TABLE ITAB2 INTO WA2 WITH KEY K = WA1-K BINARY SEARCH.

IF SY-SUBRC = 0.

ADD: WA1-VAL1 TO WA2-VAL1,

WA1-VAL2 TO WA2-VAL2.

MODIFY ITAB2 FROM WA2 INDEX SY-TABIX TRANSPORTING VAL1 VAL2.

ELSE.

INSERT WA1 INTO ITAB2 INDEX SY-TABIX.

ENDIF.

ENDLOOP.

***acceptable:***

\* TAB1: 1000 entries, Line width: 32

\* 500 entries have different keys

LOOP AT ITAB1 INTO WA.

COLLECT WA INTO ITAB2.

ENDLOOP.

SORT ITAB2 BY K.

* 如果要将一个表的记录追加到另一个表中，请使用"APPEND LINES OF itab1 TO itab2" .

**Example:**

***Unacceptable:***

\* Entries: 500 (ITAB1), 500 (ITAB2)

\* Line width: 500

\* ITAB1 is appended line by line to ITAB2.

LOOP AT ITAB1 INTO WA.

APPEND WA TO ITAB2.

ENDLOOP.

***acceptable:***

\* Entries: 500 (ITAB1), 500 (ITAB2)

\* Line width: 500

\* ITAB1 is appended in one step to ITAB2.

APPEND LINES OF ITAB1 TO ITAB2.

* 复制内表可以像其他数据类型一样使用MOVE语句。

**Example:**

***Unacceptable:***

\* Entries: 100, Line width 100

REFRESH ITAB2.

LOOP AT ITAB1 INTO WA.

APPEND WA TO ITAB2.

ENDLOOP.

***acceptable:***

\* Entries: 100, Line width 100

ITAB2[] = ITAB1[].

* 对于sort语句，请指定排序的字段。

**Example:**

***Unacceptable:***

\* Entries: 100, Line width: 500

\* Key width: 492

SORT ITAB.

***acceptable:***

\* Entries: 100, Line width: 500

\* Key width: 20

SORT ITAB BY K.

## 其他语句

### CASE语句

CASE 相对于IF语句更清晰易读，性能也稍好一些。

**Example:**

***Unacceptable:***

DATA C TYPE C.

IF C = 'A'. WRITE '1'.

ELSEIF C = 'B'. WRITE '2'.

ELSEIF C = 'C'. WRITE '3'.

ELSEIF C = 'D'. WRITE '4'.

ELSEIF C = 'E'. WRITE '5'.

ELSEIF C = 'F'. WRITE '6'.

ELSEIF C = 'G'. WRITE '7'.

ELSEIF C = 'H'. WRITE '8'.

ENDIF.

***acceptable:***

DATA C TYPE C.

CASE C.

WHEN 'A'. WRITE '1'.

WHEN 'B'. WRITE '2'.

WHEN 'C'. WRITE '3'.

WHEN 'D'. WRITE '4'.

WHEN 'E'. WRITE '5'.

WHEN 'F'. WRITE '6'.

WHEN 'G'. WRITE '7'.

WHEN 'H'. WRITE '8'.

ENDCASE.

### WHILE语句

如果可以使用WHILE语句来替换DO+EXIT语句，那么请使用WHILE语句，因为WHILE语句更易读且性能也要好一些。

**Example:**

***Unacceptable:***

DATA C TYPE C. DATA I TYPE I.

I = 0.

DO.

IF C NE SPACE. EXIT. ENDIF.

ADD 1 TO I.

IF I GT 10. C = 'X'. ENDIF.

ENDDO.

***Acceptable:***

DATA C TYPE C. DATA I TYPE I.

I = 0.

WHILE C = SPACE.

ADD 1 TO I.

IF I GT 10. C = 'X'. ENDIF.

ENDWHILE.

### 比较语句

比较语句中的两操作数尽量使用相同的数据类型，比如字符与字符的比较性能比字符与数字的比较更好。

**Example:**

***Unacceptable:***

DATA C(8) TYPE C.

IF 12345678 = C.

ENDIF.

***Acceptable:***

DATA C(8) TYPE C.

IF '12345678' = C.

ENDIF.

# 附录

## SAP业务模块命名表

|  |  |
| --- | --- |
| <XX> | 业务模块 |
| FIN | 财务模块 |
| MM | 物资管理模块 |
| SD | 销售分销模块 |
| PS | 项目管理模块 |
| PM | 设备管理模块 |
| QM | 质检模块 |
| OM | 运行模块 |
| HR | 人资模块 |
| BC | 基础模块 |
| CA | 跨模块应用 |
| SRM | SRM系统 |

## 数据字典对象类型命名表

|  |  |
| --- | --- |
| <T> | 类型 |
| T | 透明表 |
| S | 结构 |
| E | 数据元素 |
| D | 域 |

## 程序类型命名表

|  |  |
| --- | --- |
| <Y> | 程序类型代码 |
| R | 报表程序 |
| P | 对话程序 |
| F | FORM/SMARTFORM打印程序 |
| C | 数据转换程序 |
| A | 组件程序 |
| I | 公共Include程序 |