Using the Object-Generic-Subp Model

This section describes the Object-Generic-Subp model, which generates a business service (a *wrapper* subprogram) associated with up to 10 subprograms and 20 methods. The following topics are covered:

- Introduction
- Parameters for the Object-Generic-Subp Model
- User Exits for the Object-Generic-Subp Model

Note:
For more information about object-oriented development, see Overview of Object-Oriented Development.

Introduction

The Object-Generic-Subp model:

- Creates a subroutine for each subprogram used by the business service (maximum of 10 subprograms)

  Note:
  You can create additional subroutines within user exits to perform specialized functions.

- Creates methods to call the subroutines and user exits (maximum of 20 methods).

Parameters for the Object-Generic-Subp Model

The Object-Generic-Subp model has two specification panels: Standard Parameters and Additional Parameters. This section describes these panels. The following topics are covered:

- Standard Parameters Panel
- Additional Parameters Panel

Standard Parameters Panel

The following example shows the first specification panel, the Standard Parameters panel:

```
CUOGMA                  OBJECT-GENERIC-SUBP Subprogram                  CUOGMA0
Aug 20                       Standard Parameters                         1 of 2

 Module ............. NEW_____
 System ............. BIZDEMO_____________________________

 Title ............... Generic Business Service_
 Description ........ This subprogram is used to maintain the generic________
                      business service ........_____________________________
                      ____________________________________________________
                      ____________________________________________________

 Message numbers .... _  Categorize parameters _

 Subprograms
 _________ *     _________ *     _________ *     _________ *     _________ *
 _________ *     _________ *     _________ *     _________ *     _________ *




Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
right help  retrn quit                                          right main
```

The fields in the upper portion of this panel are similar for all models. For a description of these fields, see Common Fields on the Standard Parameters Panel.

Note:
If you select the Categorize Parameters option, the PARAMETER-DATA user exit is required. For information on this exit, see PARAMETER-DATA User Exit. For information on categorizing parameters, see Categorize Parameters.

The subprograms listed in Subprograms in the lower portion of this panel:

- Should have no screen I/O or navigation functionality (i.e., they cannot contain INPUT, WRITE, PRINT, DISPLAY, REINPUT statements or manage PF key functions)
- Must have at least one parameter

You can specify the names of up to 10 subprograms; you must specify at least one subprogram name.

Additional Parameters Panel

The following example shows the second specification panel, the Additional Parameters panel:

```
CUOGMB                  OBJECT-GENERIC-SUBP Subprogram                  CUOGMB0
Aug 20                      Additional Parameters                        2 of 2

  Method ... 1  ____________________________________ of 20

   Before                                              After
   Code    Order   Subroutine                          Code
     _       __    CALLNAT-ACUSTN____________________   _
     _       __    CALLNAT-BCUSTN____________________   _









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      help  retrn quit                    bkwrd frwrd       left  userX main
```

Use this panel to name the methods used by your business service and to indicate the functionality of each method (i.e., which subprogram to execute and the order of execution for the subprograms specified on the Standard Parameters panel and wrapped in the subroutines listed). In addition:

- You must define at least one method
- A subprogram is part of a method if an order number is assigned to it
- Each order (sequence) number must be unique; you cannot use the same number more than once
- You cannot mark the Before Code or After Code fields unless an order (sequence) number is specified
- If a level 1 parameter grouping is in more than one subprogram, the first one encountered is the one that is used

User Exits for the Object-Generic-Subp Model

The following example shows the User Exits panel for the Object-Generic-Subp model:

```
CSGSAMPL               OBJECT-GENERIC-SUBP Subprogram                  CSGSM0
Aug 19                           User Exits                            1 of 1

              User Exits             Exists    Sample   Required Conditional
   -------------------------------- -------- ---------- -------- -----------
 _  CHANGE-HISTORY                             Subprogram
 _  PARAMETER-DATA                             Subprogram    X          X
 _  PARAMETER-DATA-UNCATEGORIZED                                        X
 _  LOCAL-DATA
 _  MOVE-TO
 _  MOVE-TO-UNCATEGORIZED                                               X
 _  UNDEFINED-METHOD
 _  BEFORE-CODE                                Subprogram               X
 _  AFTER-CODE                                 Subprogram               X
 _  MATERIALIZE-XARRAY-PDA-TO-LDA                                        X
 _  MATERIALIZE-XARRAY-LDA-TO-PDA                                        X
 _  RESET-TEMP-MATERIALIZED                                             X
 _  MOVE-BACK
 _  MOVE-BACK-UNCATEGORIZED                                             X
 _  MISC-SUBROUTINES

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      help  retrn quit                   bkwrd frwrd
```

Notes:

1. For information about the standard user exits, see *User Exits for the Generation Models*, *Natural Construct Generation*.
2. For information about the User Exit editor, see *User Exit Editor*, *Natural Construct Generation*.

The following user exits are either required by or specific to the Object-Generic-Subp model:

- AFTER-CODE User Exit
- BEFORE-CODE User Exit
- MATERIALIZE-XARRAY-LDA-TO-PDA User Exit
- MATERIALIZE-XARRAY-PDA-TO-LDA User Exit
- MOVE-BACK User Exit
- MOVE-BACK-UNCATEGORIZED User Exit
- MOVE-TO User Exit
- MOVE-TO-UNCATEGORIZED User Exit
- PARAMETER-DATA User Exit
- PARAMETER-DATA-UNCATEGORIZED User Exit
- RESET-TEMP-MATERIALIZED User Exit

- UNDEFINED-METHOD User Exit

AFTER-CODE User Exit

The code in this exit is executed after all subprograms that had the After Code option selected on the Additional Parameters panel have been executed. If method-specific code is required, you can add it based on the value of +METHOD (indicates which business service method is executed).

To only execute the portion of the code associated with a specific subprogram, ensure that the appropriate code is specified in VALUE in the DECIDE clause associated with the corresponding subroutine (i.e., only the code in the VALUE "CALLNAT-GCDN" clause will be executed when the CALLNAT-GCDN subroutine invokes it). For example, the CALLNAT-GCDN subroutine contains a CALLNAT to the GCDN subprogram and this code is executed after that CALLNAT.

The following example shows code in the AFTER-CODE user exit for the BNUM subprogram in the SYSCSTDE library:

```
DEFINE EXIT AFTER-CODE
** Note +METHOD can also be used to
**      determine lines of execution
**      e.g. IF +METHOD = ... THEN
  DECIDE ON FIRST VALUE OF #SUBROUTINE-NAME
    VALUE "CALLNAT-GCDN"
      IF +METHOD = 'SolutionWithLowerNumbers'
/*       Lower the first number by the GCD
        #FUNCTION := 'Divide'
        INPUT-DATA.#SECOND-NUM := GCD-DATA.#RESULT
        PERFORM CALLNAT-CALC
/*    Instead of using temporary variables; temporarily used
/*    exposed field variables
        #BIZ-INPUT-OUTPUTS.#FIRST-NUM := OUTPUT-DATA.#RESULT
/*       Lower the second number by the GCD
        INPUT-DATA.#FIRST-NUM :=
          #BIZ-INPUT-OUTPUTS.#SECOND-NUM
        INPUT-DATA.#SECOND-NUM := GCD-DATA.#RESULT
        PERFORM CALLNAT-CALC
        #BIZ-INPUT-OUTPUTS.#SECOND-NUM := OUTPUT-DATA.#RESULT
/*      Move results to Calc input again to do actual division
/*      of reduced numbers
        MOVE BY NAME #BIZ-INPUT-OUTPUTS TO INPUT-DATA
      END-IF
      IF +METHOD = 'GreatestCommonDenominator'
        IF GCD-DATA.#RESULT > 1 THEN
          OUTPUT-DATA.#SUCCESS := TRUE
        ELSE
          OUTPUT-DATA.#SUCCESS := FALSE
        END-IF
      END-IF
    NONE
      IGNORE
  END-DECIDE
END-EXIT
```

BEFORE-CODE User Exit

The code in this exit is similar to the code in the AFTER-CODE user exit except it is executed before the corresponding subroutine is executed. If method-specific code is required, you can add it based on the value of +METHOD (indicates which business service method is executed).

The following example shows code in the BEFORE-CODE user exit for the BSTRINGN subprogram in the SYSCSTDE library:

```
DEFINE EXIT BEFORE-CODE
** Note +METHOD can also be used to
**      determine lines of execution
**      e.g. IF +METHOD = ... THEN
  DECIDE ON FIRST VALUE OF #SUBROUTINE-NAME
    VALUE "CALLNAT-CSUCASE" /*  U=Upper, L=Lower, M=Mixed Case
      DECIDE ON FIRST VALUE OF +METHOD
        VALUE 'ConvertToUpperCase'
          CSACASE.#FUNCTION := 'U'
        VALUE 'ConvertToLowerCase'
          CSACASE.#FUNCTION := 'L'
        VALUE 'ConvertToMixedCase'
          CSACASE.#FUNCTION := 'M'
        ANY
          EXAMINE FULL #BIZ-INPUT-OUTPUTS.#STRING FOR ' '
            GIVING LENGTH IN #BIZ-INPUT-OUTPUTS.STRING-LENGTH
        NONE
          IGNORE
        END-DECIDE
      IGNORE
    NONE
      IGNORE
  END-DECIDE
END-EXIT
```

MATERIALIZE-XARRAY-LDA-TO-PDA User Exit

This exit is used when you add X-array fields to the object generic PDA. It is used in conjunction with the MATERIALIZE-XARRAY-PDA-TO-LDA and RESET-TEMP-MATERIALIZED user exits. The code in this exit prepares for a MOVE BY NAME from a local data area (LDA) containing X-arrays to a parameter data area (PDA) containing similar X-arrays. This code temporarily resizes X-arrays before performing the MOVE BY NAME to the PDA.

These exits are only required when an X-array parameter is added to the object generic PDA (in one of the PARAMETER-DATA exits) and has not been included in the supplied subprograms. This code eliminates runtime errors when X-arrays have not been sized before a MOVE BY NAME is performed.

Tip:
To use these exits, refer to the code preceding the exits that was generated for known X-arrays.

MATERIALIZE-XARRAY-PDA-TO-LDA User Exit

This exit is used when you add X-array fields to the object generic PDA. It is used in conjunction with the MATERIALIZE-XARRAY-LDA-TO-PDA and RESET-TEMP-MATERIALIZED user exits. The code in this exit prepares for a MOVE BY NAME from a parameter data area (PDA) containing X-arrays to a local data area (LDA) containing similar X-arrays.

For more information, see MATERIALIZE-XARRAY-LDA-TO-PDA User Exit.

MOVE-BACK User Exit

This exit is used in conjunction with the MOVE-TO user exit and the PARAMETER-DATA user exit, which contains the data that is exposed to the client from the object generic subprogram. After the internal subprograms have been invoked, the data must be exposed via the parameter data area (PDA). The local variables are moved to the parameter variables in the MOVE-BACK user exit.

For more information, see PARAMETER-DATA User Exit.

MOVE-BACK-UNCATEGORIZED User Exit

This exit is used in conjunction with the MOVE-TO-UNCATEGORIZED user exit and the PARAMETER-DATA-UNCATEGORIZED user exit, which contains the data that is exposed to the client from the object generic subprogram. After the internal subprograms have been invoked, the data must be exposed via the parameter data area (PDA). The local variables are moved to the parameter variables in the MOVE-BACK-UNCATEGORIZED user exit.

For more information, see PARAMETER-DATA-UNCATEGORIZED User Exit.

MOVE-TO User Exit

This exit is used in conjunction with the MOVE-BACK user exit and the PARAMETER-DATA user exit, which contains the data that is exposed to the client from the object generic subprogram. To pass this data to subprograms, the data must be moved to local data areas in the MOVE-TO user exit.

For more information, see PARAMETER-DATA User Exit.

MOVE-TO-UNCATEGORIZED User Exit

This exit is used in conjunction with the MOVE-BACK-UNCATEGORIZED user exit and the PARAMETER-DATA-UNCATEGORIZED user exit, which contains the data that is exposed to the client from the object generic subprogram. To pass this data to subprograms, the data must be moved to local data areas in the MOVE-TO-UNCATEGORIZED user exit.

For more information, see PARAMETER-DATA-UNCATEGORIZED User Exit.

PARAMETER-DATA User Exit

The PARAMETER-DATA user exit is required if you specified the Categorize Parameters option on the Standard Parameters panel. This exit is used in conjunction with two other exits: MOVE-TO and MOVE-BACK. (For more information on categorizing parameters, see Categorize Parameters.)

The object generic subprogram wraps up to 10 subprograms into one subprogram. The PARAMETER-DATA user exit contains the data that is exposed to the client from the object generic subprogram. To pass this data to the subprograms, it must be moved to local data areas in the MOVE-TO user exit. Similarly, after the internal subprograms have been invoked, the data must be exposed via the parameter data area (PDA). The local variables are moved to the parameter variables in the MOVE-BACK user exit.

The PARAMETER-DATA user exit allows the user to choose which level 1 parameter groupings will be input, input-output, state, and output. The same parameter name cannot be listed under the same input, input-output, state, or output groupings. If this occurs, you must revise the generated code.

To help select the level 1 parameter groupings, the following panel is displayed when you press Enter on the User Exits panel for the Object-Generic-Subp model:

```
CUOGMC                      Natural Construct                          CUOGMC0
Nov 16         Object-Generic-Subp Subprogram Build Report              1 of 1

 1__            Level Ones             Input  Input-Output  State  Output
    --------------------------------  -----  ------------  -----  ------
 1   ACUSTNK                           _          _         _      _
 2   ACUSTND                           _          _         _      _
 3   ACUSTNP                           _          _         _      _
 4   CDBRPDA                           _          _         _      _
 5   MSG-INFO                          _          _         _      _
 6   BCUSTE1                           _          _         _      _
 7   CDBUPDA                           _          _         _      _
 8   CDBUINFO                          _          _         _      _
 9   BUSINESS-INFO                     _          _         _      _









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      help  retrn             gen         bkwrd frwrd
```

Based on the grouping, data is moved from the exposed PDAs to the internal LDAs used for the subprograms. You can define up to 100 level 1 parameter groupings. Up to four unique PDAs can be duplicated across the subprograms.

Note:
While using the NCSTBGEN command to regenerate multiple modules in batch mode, object generic subprograms may not be regenerated. For example, if the parameters have been categorized (i.e., defined within user exits), you must regenerate the PARAMETER-DATA user exit from the client.

Structure of the Generated Code

The following example shows the skeleton view of code generated by the PARAMETER-DATA user exit:

```
DEFINE DATA
PARAMETER
1 #INPUT
…
1 #INPUT-OUTPUT
…
1 #STATE
…
1 #OUTPUT
LDAs
END-DEFINE
MOVE BY NAME Pdas to Ldas
```

```
** SAG EXIT POINT AFTER-PDA-TO-LDA-MOVE
DECIDE ON FIRST VALUE OF +METHOD
VALUE 'ABC'
  EXECUTE-BEFORE := TRUE (optional)
  EXECUTE-AFTER := TRUE (optional)
  PERFORM nnnn2-CALLNAT
  EXECUTE-AFTER := TRUE (optional)
  PERFORM nnnn1-CALLNAT
VALUE 'DEF'
  EXECUTE-BEFORE := TRUE (optional)
  EXECUTE-AFTER := TRUE (optional)
  PERFORM nnnn3-CALLNAT
  EXECUTE-AFTER := TRUE (optional)
  PERFORM nnnn1-CALLNAT
END-DECIDE
MOVE BY NAME LDAS to PDAs
** SAG EXIT POINT AFTER-LDA-TO-PDA-MOVE
*
DEFINE SUBROUTINE nnnn1-CALLNAT
  SUBROUTINE-NAME := 'nnnn1-CALLNAT'
  IF EXECUTE-BEFORE THEN
     PERFORM BEFORE
  END-IF
  CALLNAT 'nnnn1' ....
  IF EXECUTE-AFTER THEN
     PERFORM AFTER
  END-IF
  RESET EXECUTE-BEFORE EXECUTE-AFTER
END-SUBROUTINE
*
DEFINE SUBROUTINE nnnn2-CALLNAT
  SUBROUTINE-NAME := 'nnnn1-CALLNAT'
  IF EXECUTE-BEFORE THEN
     PERFORM BEFORE
  END-IF
  CALLNAT 'nnnn1' ....
  IF EXECUTE-AFTER THEN
     PERFORM AFTER
  END-IF
  RESET EXECUTE-BEFORE EXECUTE-AFTER
END-SUBROUTINE
*
DEFINE SUBROUTINE nnnn3-CALLNAT
  SUBROUTINE-NAME := 'nnnn1-CALLNAT'
  IF EXECUTE-BEFORE THEN
     PERFORM BEFORE
  END-IF
  CALLNAT 'nnnn1' ....
  IF EXECUTE-AFTER THEN
     PERFORM AFTER
  END-IF
  RESET EXECUTE-BEFORE EXECUTE-AFTER
END-SUBROUTINE
*
DEFINE SUBROUTINE BEFORE
    EXECUTE-BEFORE := FALSE
** User Exit BEFORE Code
* Note that +METHOD can also be used in this logic
  DECIDE ON FIRST VALUE OF SUBROUTINE-NAME
     VALUE 'nnnn1-CALLNAT'
        IGNORE
     VALUE 'nnnn2-CALLNAT'
```

```
        IGNORE
      NONE
         IGNORE
   END-DECIDE
** User Exit End code
  ESCAPE ROUTINE
END-SUBROUTINE
*
DEFINE SUBROUTINE AFTER
  EXECUTE-AFTER := FALSE
** User Exit AFTER Begin Code
* Note that +METHOD can also be used in this logic
  DECIDE ON FIRST VALUE OF SUBROUTINE-NAME
     VALUE 'nnnn1-CALLNAT'
        IGNORE
     VALUE 'nnnn2-CALLNAT'
        IGNORE
      NONE
         IGNORE
   END-DECIDE
** User Exit End code
  ESCAPE ROUTINE
END-SUBROUTINE
```

PARAMETER-DATA-UNCATEGORIZED User Exit

Use this exit if you want to expose more parameters to the client than are found in the specified subprograms and you did not specify the Categorize Parameters option on the Standard Parameters panel (for example, you can use this exit to expose a message field if the specified subprograms do not have one). This exit is optional and is used in conjunction with the MOVE-TO-UNCATEGORIZED and MOVE-BACK-UNCATEGORIZED user exits. The MOVE-TO-UNCATEGORIZED and MOVE-FROM-UNCATEGORIZED user exits are similar to the MOVE-TO and MOVE-FROM exits for the PARAMETER-DATA user exit except they are used when the Categorize Parameters option is not selected.

If you decide not to categorize parameters, every PDA from the specified subprograms will be exposed to the client. The subprogram created by the object generic subprogram will have two types of variables: parameters that will become the parameters of the business service and local data that will become the parameters to the supplied subprograms.

Initially, code is automatically generated into the MOVE-TO and MOVE-FROM exits when the Categorize Parameters option is specified. This does not happen when the option is not selected, as more code can be generated outside of user exits.

RESET-TEMP-MATERIALIZED User Exit

The code in this exit temporarily resizes X-arrays before performing the MOVE BY NAME to the LDA or PDA. Use this exit when you add X-array fields to the object generic PDA.

This exit is used in conjunction with the MATERIALIZE-XARRAY-LDA-TO-PDA and MATERIALIZE-XARRAY-PDA-TO-LDA user exits. For more information, see MATERIALIZE-XARRAY-LDA-TO-PDA User Exit.

UNDEFINED-METHOD User Exit

The code in this exit determines what happens when an undefined method is added to the object generic subprogram and has not been included in the specifications.

Note:
In general, the repository should access the same methods as the object generic code. If not, use this exit to define the new methods.