CST-Validate Model

This section describes the CST-Validate model, which is used to create the validation subprogram for a model. The validation subprogram verifies inputs for the model during the generation process.

This section covers the following topics:

- Introduction
- Parameters for the CST-Validate Model
- User Exits for the CST-Validate Model

Introduction

If you code validations within the maintenance panel modules, it is difficult to invoke the validations from batch programs or GUI clients. Instead, you can consolidate all model validation within a validation subprogram. To confirm input values for your model, use the CST-Validate model to generate a validation subprogram and then add the subprogram to the model record on the Maintain Models panel.

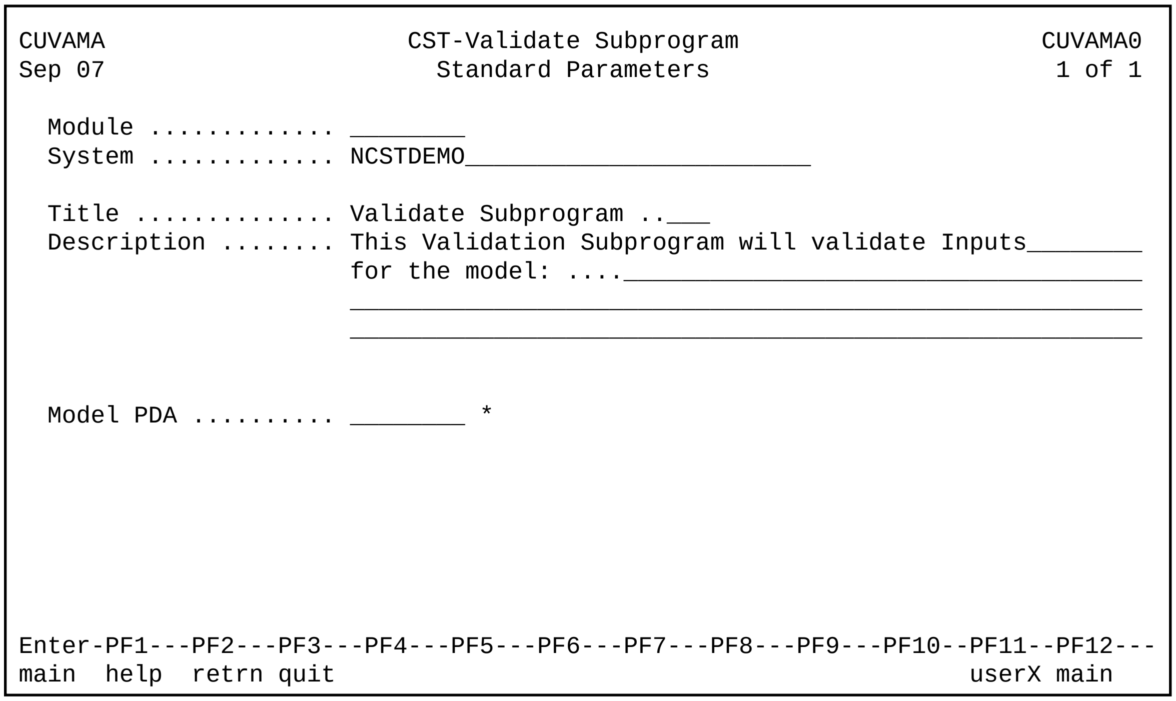
The following example shows how to use a validation subprogram to validate inputs for a maintenance panel:

```
**SAG DEFINE EXIT VALIDATE-DATA
  ASSIGN CSAVAL.VALIDATE-SPECIFIC-FIELD(1) = ’field1’
  ASSIGN CSAVAL.VALIDATE-SPECIFIC-FIELD(2) = ’field2’
  ASSIGN CSAVAL.VALIDATE-SPECIFIC-FIELD(3) = ’field3’
  CALLNAT  ’CUBOVAL’  CSAVAL
                      CUBOPDA    /*your model PDA name
                      CU—PDA
                      CSAMARK
                      CSAERR
                      CSASTD
  PERFORM  REINPUT-MESSAGE
*
**SAG  END-EXIT
```

Parameters for the CST-Validate Model

Use the CST-Validate model to create the validation subprogram. This model has one specification panel, Standard Parameters.

Standard Parameters Panel



```
CUVAMA                    CST-Validate Subprogram                    CUVAMA0
Sep 07                      Standard Parameters                       1 of 1

  Module ............. ________
  System ............. NCSTDEMO________________________

  Title .............. Validate Subprogram ..___
  Description ........ This Validation Subprogram will validate Inputs________
                       for the model: ....____________________________________
                       _______________________________________________________
                       _______________________________________________________


  Model PDA .......... ________ *







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

The input fields on the Standard Parameters panel are:

Field	Description
Module	Name specified on the Generation main menu. The name of the validation subprogram must be alphanumeric and no more than eight characters in length. Use the following naming convention: `CX`*`xx`*`VAL` where *xx* uniquely identifies your model.
System	Name of the system (by default, the name of the current library). The system name must be alphanumeric, not exceed 32 characters in length, and does not have to be associated with a Natural library ID. (The combination of the module name and system name is used as a key to access help information for the generated subprogram.)
Title	Title for the generated subprogram. The title identifies the subprogram for the List Generated Modules function on the Generation main menu and is used internally for program documentation.
Description	Brief description of the subprogram. The description is inserted in the banner at the beginning of the subprogram and is used internally for program documentation.
Model PDA	Name of the PDA used by the model for which you are generating the validation subprogram.

User Exits for the CST-Validate Model

```
CSGSAMPL                      Natural Construct                         CSGSM0
Sep 07                            User Exits                             1 of 1
               User Exit              Exists    Sample   Required Conditional
    ------------------------------- -------- ---------- -------- ------------
 _  CHANGE-HISTORY                            Subprogram
 _  LOCAL-DATA
 _  GENERATE-VALIDATIONS
 _  GENERATE-SUBROUTINES                      Subprogram
```

For information about these user exits, see Supplied User Exits. For information about using the User Exit editor, see *User Exit Editor*, *Natural Construct Generation*.

Code Validations

The CST-Validate model codes validations as subroutines in the GENERATE-SUBROUTINES user exit. For each #PDAX-FIELD-NAME field you want to validate, create a subroutine called V-*field-name* to perform the validations. Whenever a validation error is found, the V-*field-name* subroutine must:

- Assign `CSASTD.RETURN-CODE = ‘E’`
- Assign the error message in CSASTD.MSG
- Perform an ESCAPE-ROUTINE to bypass subsequent checks

Notes:

1. To retrieve SYSERR messages, use the CU--VERR copycode.
2. For more information about coding validations, see GENERATE-SUBROUTINES.

Validate Array Fields

For array fields, the V-*field-name* subroutine validates all occurrences for which validation is requested. These occurrences are supplied in the #INDEX.#FROM (1:3) fields (redefined into #I1, #I2 and #I3). To return multiple errors (for separate field occurrences), perform the CHECK-AFTER-EDIT subroutine when an error occurs within an array field. This will add the error to the error list but allow editing of subsequent indexes to occur.

The following example shows the validation routine for a two-dimensional array called #PDAX-PHYSICAL-KEY:

```
**********************************************************************
DEFINE SUBROUTINE V_PHYSICAL-KEY
**********************************************************************
*
  FOR #INDEX.#OCC(1) = #INDEX.#FROM(1) TO #INDEX.#THRU(1)
    FOR #INDEX.#OCC(2) = #INDEX.#FROM(2) TO #INDEX.#THRU(2)
      /*
      /* Validate #PDAX-PHYSICAL-KEY(#I1,#I2)
      ASSIGN CPAEL.FILE-NAME = CUBOPDA.#PDAX-PRIME-FILE
      ASSIGN CPAEL.FILE-CODE = CUBOPDA.#PDAX-PHYSICAL-KEY(#I1,#I2)
      ASSIGN CPAEL.DDM-PREFIX = CPAFI.DDM-PREFIX
      CALLNAT 'CPUEL' CPAEL CSASTD
      IF NOT CPAEL.#FIELD-FOUND
        ASSIGN CNAMSG.MSG-DATA(1) = CPAEL.FIELD-NAME
        ASSIGN CNAMSG.MSG-DATA(3) = CPAEL.FILE-NAME
        INCLUDE CU--VER2 '0096'
            ''':1::2:not in:3:'''
            'CUBOPDA.#PDAX-PHYSICAL-KEY(#I1,#I2)'
      END-IF
    END-FOR
  END-FOR
END-SUBROUTINE /* V_PHYSICAL-KEY
```

Tips

- If you do not want to exit the current subroutine, as with array processing, use the CU--VERZ copycode instead of CU--VERR.
- To return a warning message, rather than an error, use the CU--VWAR copycode.