FLM Developer Guide

ABAP/4 Developments

Author: Chris Scott
File name: 260 Developer Guide
Version: 1
Distribution: Project team

Version History

Version	Author(s)	Reason for update
1	Chris Scott	Initial release of the document

Table of Contents

1 Form User-exits

Form-level user exits are accessed via transaction /FLM/FORM_MANAGER.
All user-exits are available to all form types; there is no dependency on settings selected in the New Form Wizard.

1.1 Form-level pre-population

The following data is available within pre-population user-exits:

<g_data> Internal table of type /FLM/XML_TAB_T storing all current form data and one instance of each field.

<g_ccode> 3-character customer code.

<g_ftype> 4-character form type.

<g_doc> 10-character document number if passed in. This is used for the offline form scenario triggered by application document output.

<g_user> The user id. For offline processing this is the user determined by /FLM/CORE->GET_OFFLINE_USER (stored on the customer code table.) For online processing this is the user id passed in by FLM Portal.

The form level pre-population user-exit can be used to prepopulate any field on the form. It can also be used to create instances of repeating subforms to pre-populate fields within row data – or nested subforms down to 3 levels of nesting.

The return parameter is 'ex_data', which is an internal table of type /FLM/XML_TAB_T. This parameter is set to equal <g_data> before the user-exit is called.

It is easier to update fields that are in non-repeating subforms with a field user-exit, as in this case there is no need to handle the internal table.

However, if the same table selections or programming logic is required to determine several form fields, then it is easier to use the logic just once, at the form level.

The syntax for updating fields in non-repeating subforms is:

```
READ TABLE ex_data WITH KEY name = 'EBELN' INTO l_data.
IF sy-subrc eq 0.
  MOVE wa_header-ekko-ebeln TO l_data-value.
  MODIFY ex_data INDEX sy-tabix FROM l_data.
ENDIF.
```

The syntax for filling fields within a repeating subform called ‘ITEM’ from an internal table ‘itab’ is:

```
 DATA:          l_index              TYPE sytabix,
                l_item_row(3)        TYPE n,
                l_item_subform       TYPE string,
                l_data_value         TYPE string,
                l_data               TYPE /flm/xml_tab.
*
 FIELD-SYMBOLS:
                 <itab>              TYPE itab.
*
LOOP AT itab ASSIGNING <itab>.
   l_item_row = sy-tabix.
*
   move <itab>-ebelp to l_data_value.
   CALL METHOD /flm/sfs=>field_prepopulate
     EXPORTING
       im_data      = ex_data
       im_subform = 'ITEM'                    “Subform name
       im_row       = l_item_row
       im_field     = 'EBELP'                 “Field name
       im_value     = l_data_value
     IMPORTING
       ex_data      = ex_data.
*
         ...
ENDLOOP.
```

1.2 E-mail address derivation

This user-exit returns an internal table of e-mail addresses and is used when an offline form is required to be dispatched to multiple recipients.

The following import parameters are available:

- IM_EMAIL_STAT-CCODE Customer code
- IM_EMAIL_STAT-FTYPE Form type
- IM_EMAIL_STAT-FLANG Language
- IM_EMAIL_STAT-FVER Form version
- IM_EMAIL_STAT-STAT_IN Form status
- IM_EMAIL_STAT-RECEIV_ADDR Receiver's e-mail address
- IM_EMAIL_STAT-EMAIL_TITLE E-mail title text
- IM_EMAIL_STAT-EMAIL_BODY E-mail body text
- IM_EMAIL_STAT-EMAIL_ATT_NAME E-mail attachment text
- IM_DOCUMENT Application document number

The export parameter is EX_EMAIL_ADDRS which is an internal table with structure /FLM/EMAIL. We can update the following fields only within this structure:

- RECEIV_ADDR Receiver's e-mail address
- EMAIL_TITLE E-mail title text
- EMAIL_BODY E-mail body text
- EMAIL_ATT_NAME E-mail attachment text

In this user-exit we can read the document data (using the document number) to find the partner number and then read the e-mail address from the partner's address details. The syntax required is:

```
Data:  wa_email      TYPE /flm/email,
       l_smtp_addr   TYPE ad_smtpadr.

call method /flm/sfs-> GET_PARTNER_ADDR_SMTP
   EXPORTING
     im_parvw       = im_parvw
     im_parnr       = im_parnr
   IMPORTING
     ex_smtp_addr = l_smtp_addr.

Wa_email = IM_EMAIL_STAT.
wa_email-RECEIV_ADDR = l_smtp_addr
APPEND wa_email TO ex_email_addrs.
```

Note: In FLM version 2.4 we cannot pass in the partner from the NAST record for offline forms. Instead we need to start with document data in the user-exit.

1.3 Workflow (FLM Routing Server)

The workflow user-exit can be used to determine the subsequent form owner, version, status and set flags to trigger the sending of an offline form or notification e-mail in the case of online forms.

The following import parameters are available:

- im_action Action
- im_instance Form instance (contains form type, form id etc)
- im_ftransport Online/Offline flag
- im_owner Form owner
- im_remind E-mail notification flag
- im_status Form status

The following export parameters are available:

- ex_owner New owner
- ex_ftransport Online/Offline flag
- ex_fver New version
- ex_remind E-mail notification flag
- ex_status New status

Typically the logic for determining the new workflow options will be driven by custom tables or by navigating the HR organisational structure.

1.4 Version

A new version can be determined prior to form rendering using the version user-exit. There are no import parameters in FLM version 2.4, so the version can only be derived from the customer code, form type and system variables/constants/TVARV variables etc. The export parameter is ex_version.

1.5 Language

A new language can be determined prior to form rendering using the language user-exit. There are no import parameters in FLM version 2.4, so the language can only be derived from the customer code, form type and system variables/constants/TVARV variables etc. The export parameter is ex_lang.

2 Field User-exits

The following data is available within all field-level user-exits:

<g_data> Internal table of type /FLM/XML_TAB_T storing all current form data and one instance of each field.

<g_ccode> 3-character customer code.

<g_ftype> 4-character form type.

<g_field> The name of the currently selected field.

2.1 Field-level prepopulation

In addition to the core data, the following fields are available:

<g_value> The value of the currently selected field.

<g_doc> 10-character document number if passed in. This is used for the offline form scenario triggered by application document output.

<g_user> The user id. For offline processing this is the user determined by /FLM/CORE->GET_OFFLINE_USER (stored on the customer code table.) For online processing this is the user id passed in by FLM Portal.

The export parameter is ex_value, which has type 'string'.

2.2 Possible entries

In addition to the core data, the following fields are available:

<g_doc> 10-character document number if passed in. This is used for the offline form scenario triggered by application document output.

<g_user> The user id. For offline processing this is the user determined by /FLM/CORE->GET_OFFLINE_USER (stored on the customer code table.) For online processing this is the user id passed in by FLM Portal.

The export parameter is ex_form_data, which is an internal table with two fields, name and value.

Note: In FLM version 2.4, the derived data value needs to be written to the 'name' field and the data description needs to be written to the 'value' field.

The required syntax is of the form:

```
DATA: l_form_data TYPE /flm/form_data.
MOVE '0' TO l_form_data-name.
MOVE 'OFF' TO l_form_data-value.
APPEND l_form_data TO ex_form_data.
```

2.3 Derivation

In addition to the core data, the following fields are available:

<g_return> The 'return' field submitted back from the form. This has the structure: <ACTION> **+** <CUSTOMER CODE> **-** <FORM TYPE> **-** <LANGUAGE> **-** <VERSION> **-** <FORM ID> **-** <VARIANT> **+** <RECIPIENT EMAIL ADDRESS>

<g_path> The path of the currently selected field

<g_value> The value of the currently selected field

Changes to <g_value> cause the field value to change before posting.

All fields need to already exist on the form – we cannot derive a field in a new instance of a subform through the derivation user-exit.

To read values in the <g_return> field we need to split the field as follows:

```
SPLIT <g_return> AT '+' INTO l_action l_cms_doc l_rec_email.
```

Then we can split the cms document reference using the method /FLM/CORE-> SPLIT_XDP_CMS_DOC.

2.4 Substitution

In addition to the core data, the following fields are available:

<g_return> The 'return' field submitted back from the form. This has the structure: <ACTION> **+** <CUSTOMER CODE> **-** <FORM TYPE> **-** <LANGUAGE> **-** <VERSION> **-** <FORM ID> **-** <VARIANT> **+** <RECIPIENT EMAIL ADDRESS>

<g_path> The path of the currently selected field

<g_value> The value of the currently selected field

Changes to <g_value> cause the field value to change before posting.

2.5 Validation

In addition to the core data, the following fields are available:

<g_return> The 'return' field submitted back from the form. This has the structure: <ACTION> **+** <CUSTOMER CODE> **-** <FORM TYPE> **-** <LANGUAGE> **-** <VERSION> **-** <FORM ID> **-** <VARIANT> **+** <RECIPIENT EMAIL ADDRESS>

<g_path> The path of the currently selected field

<g_value> The value of the currently selected field

The export parameters are:

- ex_response — String composed of one or several of the following codes:
  - A On-Line - Error - reject form
  - B On-Line - Warning - log event
  - C Off-Line - Warning - log event
  - D Off-Line - Error - return form
  - E Off-Line - Error - delete form
- ex_mess_num — Message number from class /FLM/SFS
- ex_msgvar1 — Error variable 1
- ex_msgvar2 — Error variable 2
- ex_msgvar3 — Error variable 3
- ex_msgvar4 — Error variable 4

Note: in FLM version 2.4 the message class is always 'FLM/SFS' which is in the FLM namespace and should not be changed. It is therefore recommended to use message number 999 and pass in the validation text as one of the message variables.

The syntax should take the following form:

```
if <g_value> is INITIAL.
  ex_response = 'A'.
  ex_mess_num = '999'.
  ex_msgvar1  = 'Initial field not permitted'.
  ex_msgvar2  = <g_field>.
  ex_msgvar3  = space.
  ex_msgvar4  = space.
endif.
```

3 Offline Forms

3.1 Offline form triggered by FLM Output

The email user-exit described above is always used to determine the e-mail recipient for an offline form.

3.2 Offline form triggered for form distribution list

The email user-exit described above is always used to determine a distribution list.
The offline form will be triggered by the FLM offline submissions utility or by a custom program that calls function module /FLM/OFFLINE_FORM_SUBMIT.

3.3 Offline form triggered by FLM Routing Server

The normal email user-exit is triggered for the determination of e-mail recipients for offline forms triggered by FLM Routing configuration or user-exit.

4 Posting Adapters

4.1 Posting adapter coding

All posting adapters must have the following import parameters:

```
FORMS_DATA          TYPE  /FLM/XML_TAB_T
SEQUENCE            TYPE  /FLM/PROCESS_SEQ
DIALOGUE_MODE       TYPE  CHAR1
```

And the following export parameters:

```
POSTED_DOC          TYPE  /FLM/PDOC
RETURN              TYPE  BAPIRETURN1
NO_POST_ATTEMP      TYPE  FLAG
```

Normally the posting adapter will loop around the FORMS_DATA internal table, taking the data from the form and filling other internal tables and/or structures required as import parameters by BAPIs to make the final SAP update.

Any BAPI returns are passed back and and generated document number is also returned.

```
TYPES: t_return        TYPE TABLE OF bapireturn,

DATA:  subform_tab         TYPE /FLM/XML_TAB_T ,
       subform_wa_t        TYPE /FLM/XML_TAB_T.
       w_return            TYPE t_return,
       path_tab            TYPE TABLE OF string,
       l_path_part         TYPE string,
       l_parent_path       TYPE string,
       l_parent_path_c(80)   TYPE c,
       l_parent_path_len   TYPE i,
       l_subform(3)        TYPE n,
       t_lines             TYPE i.
```

Call method **/FLM/SFS-> DATA_ADD_PARENT_PATH** passing in FORMS_DATA and receiving back SUBFORM_TAB.

Now we have all the parent paths we can loop at this to get all the form data for a particular instance of a subform.

Use the following syntax for fields in non-repeating subforms:

```
READ TABLE forms_data ASSIGNING <f_formfld>
  WITH KEY name = 'DELIV_EXT'.
<bapi_import_structure-field> = <f_formfld>-value.
```

Use the following sytax for fields in repeating subforms:

```
* Get the first occurance of an item field:
READ TABLE subform_tab ASSIGNING <subform>
    WITH KEY name = 'MATNR'.

  l_parent_path_c = <subform>-parent_path.
  l_parent_path_len = STRLEN( l_parent_path_c ) - 3.
  l_subform = 1.

  WHILE l_subform LT 4.
    MOVE l_subform TO l_parent_path_c+l_parent_path_len(3).
    MOVE l_parent_path_c TO l_parent_path.
    CLEAR: subform_wa_t, wa_inb_del_item.
*
    LOOP AT subform_tab ASSIGNING <subform>
WHERE parent_path = l_parent_path.
      APPEND <subform> TO subform_wa_t.
    ENDLOOP.

    DESCRIBE TABLE subform_wa_t LINES t_lines.
    IF t_lines GT 0.

* Now we have a table of the fields in just one row.
      READ TABLE subform_wa_t ASSIGNING <subform>
        WITH KEY name = 'MATNR'.
      <bapi_import_item_wa-field> = <subform>-value.

READ TABLE subform_wa_t ASSIGNING <subform>
        WITH KEY name = ...
...

* Now append the item row to the BAPI import internal table parameter
APPEND <bapi_import_item_wa> TO <bapi_import_item>.
      endif.
    ELSE.
      EXIT.
    ENDIF.
    ADD 1 TO l_subform.
  ENDWHILE.
```

Once all the BAPI import parameters are filled then the BAPI is called and the results passed back to the calling program.

5 Output Forms

5.1 Output forms triggered by SAP application output

An interface with name /FLM/xx needs to be defined where xx is the SAP application code (EF = purchasing, V1 = sales order etc.)

The import parameters are always:

Parameter	Assignment	Type name	Optional flag	Pass value
/1BCDWB/DOCPARAMS	TYPE	SFPDOCPARAMS	1	1
NAST	TYPE	NAST	0	1

The export parameters are always:

Parameter	Assignment	Type name	Pass value
/1BCDWB/FORMOUTPUT	TYPE	FPFORMOUTPUT	1

Within the 'Global data' part of the interface we add the structures required to be mapped to form fields.

Within the 'Code Initialization' part of the interface, we add the code to call the function module to fill the structures defined in the global data. We export the 'NAST' table entry and import the data in the structures required to map to the form. For example:

```
CALL FUNCTION '/FLM/FLMO_EF'
  EXPORTING
    nast                = nast
  IMPORTING
    ef_po_print         = ef_po_print
  EXCEPTIONS
    data_error          = 1
    OTHERS              = 2.
```

Note that recipient e-mail addresses are derived from the partner in the condition record (passed on to table NAST)

5.2 Output forms triggered for form distribution list

For output forms to a large distribution list, use the offline form scenario with no interactive fields. Use the e-mail user-exit to determine the distribution list.

5.3 HR output forms

PDF output forms are already integrated with HR output. Use transaction HRFORMS to branch to the SAP Form Builder which has Adobe Designer embedded. There is no integration with FLM.

5.4 FI output forms

PDF output forms are already integrated with FI correspondance. Link the custom program to the form in table T001F through view V_T001F2. Then correspondance program RFKORI80 will use this table and generate PDF output forms. There is no integration with FLM.

6 Index of methods for form data handling

This section describes the other methods delivered as part of FLM that can be used for data handling in user-exits.

6.1 Get the complete address details from an address number

/FLM/SFS=> ADRNR_TO_ADDR_COMP

IM_ADRNR	TYPE ADRNR	Address number
EX_ADDR_COMPLETE	TYPE SZADR_ADDR1_COMPLETE	Partner complete address

This method reads the complete address details from an address number.

6.2 Get the complete address details from a partner number

/FLM/SFS=> GET_PARTNER_ADDR_COMP

IM_PARVW	TYPE PARVW	Partner type
IM_PARNR	TYPE PARNR	Partner number
EX_ADDR_COMPLETE	TYPE SZADR_ADDR1_COMPLETE	Partner complete address

This method reads the complete address details from a partner type and number.

6.3 Get e-mail address from partner number

/FLM/SFS=>GET_PARTNER_ADDR_SMTP

IM_PARVW	TYPE PARVW	Partner type
IM_PARNR	TYPE PARNR	Partner number
EX_SMTP_ADDR	TYPE AD_SMTPADR	Partner e-mail address

This method reads the e-mail address from a partner type and number.

6.4 Get address from address number into single text field

/FLM/SFS=>ADRNR_TO_TEXT_FIELD

I_ADRNR	TYPE ADRNR	Address number
O_ADDRESS	TYPE STRING	Formatted address

Note that in FLM version 2.4 we do not include the country in the formatted address; this method should be cloned if any address lines are required that are missing from the returned address.

6.5 Get standard text into single text field

/FLM/SFS=>READ_TEXT_TO_TEXT_FIELD

IM_TDID	TYPE TDID	Text ID
IM_SPRAS	TYPE SPRAS	Language
IM_TDNAME	TYPE TDOBNAME	Name
IM_TDOBJECT	TYPE TDOBJECT	Object
EX_TEXT	TYPE STRING	Output text

This method reads the contents of a standard text and concatenates them into a single string, adding in carriage return codes at the end of each line so that the standard text is easily formatted when bound to a form.

6.6 Prepopulate field within a subform

/FLM/SFS=>FIELD_PREPOPULATE

IM_DATA	TYPE /FLM/XML_TAB_T	Table type for XML table
IM_SUBFORM	TYPE STRING	Parent subform name
IM_ROW	TYPE INT3	Parent subform row instance
IM_FIELD	TYPE STRING	Field name
IM_VALUE	TYPE STRING	Field value
EX_DATA	TYPE /FLM/XML_TAB_T	Table type for XML table

This methods is used for repeating subform handling within form prepopulation.

6.7 Add parent paths to form data xml table

/FLM/SFS=>DATA_ADD_PARENT_PATH

IM_DATA	TYPE /FLM/XML_TAB_T	Table type for XML table
EX_DATA	TYPE /FLM/XML_TAB_T	Table type for XML table

This method is used for repeating subform handling within posting adapters.

6.8 Get HR Personnel number from user id

/FLM/SFS=>UNAME_GET_PERNR

IM_UNAME	TYPE UNAME	User Name
IM_SUBTY	TYPE SUBTY DEFAULT '0001'	Sub type
IM_DATUM	TYPE DATUM	Date
IM_OBJPS	TYPE OBJPS DEFAULT ''	Object ID
IM_SPRPS	TYPE SPRPS DEFAULT ''	Lock indicator
EX_PERNR	TYPE PERSNO	Personnel number

The link between a user name and the personnel number is stored in info type 0105, subtype 0001. The method does a simple selection on table PA0105.

6.9 Get User ID from HR Personnel number

/FLM/SFS=>PERNR_GET_UNAME

IM_PERNR	TYPE PERSNO	Personnel number
IM_SUBTY	TYPE SUBTY DEFAULT '0001'	Sub type
IM_DATUM	TYPE DATUM	Date
IM_OBJPS	TYPE OBJPS DEFAULT ''	Object ID
IM_SPRPS	TYPE SPRPS DEFAULT ''	Lock indicator
EX_UNAME	TYPE UNAME	User Name

The link between a user name and the personnel number is stored in info type 0105, subtype 0001. The method does a simple selection on table PA0105.

6.10 Get E-mail address from user id

/FLM/CORE=>GET_USER_EMAIL

IM_USER	TYPE UNAME	FLM: Form Owner
EX_EMAIL	TYPE AD_SMTPADR	FLM: Form Action

This method returns the e-mail address from a user's default data.

6.11 Get E-mail address from HR Personnel number

/FLM/SFS=>PERNR_GET_EMAIL

IM_SUBTY	TYPE SUBTY DEFAULT '0010'	Sub type
IM_DATUM	TYPE DATUM	Date
IM_OBJPS	TYPE OBJPS DEFAULT ''	Object ID
IM_SPRPS	TYPE SPRPS DEFAULT ''	Lock indicator
IM_PERNR	TYPE PERSNO	Personnel number
EX_EMAIL	TYPE /FLM/EMAIL_RECE	E-mail address

The Link between a personnel number and their e-mail address is stored in info type 0105, subtype 0010. This method performs a simple selection on table PA0105.

6.12 Navigate HR organisational structure

/FLM/SFS=>PERNR_GET_MANAGER

IM_PERNR	TYPE HROBJID	Personnel number
IM_PLVAR	TYPE PLVAR DEFAULT '10'	Plan Version
IM_DATUM	TYPE DATUM	Date
IM_PERNR_PROLE_RELAT	TYPE RELAT DEFAULT '008'	Relationship Between Objects
IM_PROLE_DEPT_RELAT	TYPE RELAT DEFAULT '003'	Relationship Between Objects
IM_DEPT_SROLE_RELAT	TYPE RELAT DEFAULT '012'	Relationship Between Objects
IM_SROLE_SPERNR_RELAT	TYPE RELAT DEFAULT '008'	Relationship Between Objects
EX_SPERNR	TYPE HROBJID	Manager Personnel number

This method performs several selections on table HRP1001 passing in relationships to find an employee's supervisor.

Note that this works only with the structure desribed below, and a check is required afterwards in case the employee passed in was a supervisor: in practise we may need to clone this method depending on the organisational structure in HR.

Dept [O]		
B003->	**Employee role [S]**	
	A008->	**Employee [P]**
B012->	**Supervisor role[S]**	
	A008->	**Supervisor [P]**

6.13 Get previous form owner

/FLM/CORE=>GET_FORM_PREV_OWNER

IM_CCODE	TYPE /FLM/CUST_CODE	SFS: Customer Code
IM_FTYPE	TYPE /FLM/FTYPE_CODE	SFS: Form Type
IM_FLANG	TYPE /FLM/FLANG	FLM: Form Language
IM_FVER	TYPE /FLM/FVER	FLM: Form Version
IM_FID	TYPE /FLM/FID	FLM: Form ID
IM_FID_VAR	TYPE /FLM/ID_VAR	FLM: Form Variant
VALUE(RE_OWNER)	TYPE XUBNAME	User Name in User Master Record

This method finds the last previous form owner. It is useful for owner derivation within workflow user-exits for rejection actions.

6.14 Get previous form actioner

/FLM/CORE=>GET_FORM_PREV_ACTIONER

IM_CCODE	TYPE /FLM/CUST_CODE	SFS: Customer Code
IM_FTYPE	TYPE /FLM/FTYPE_CODE	SFS: Form Type
IM_FLANG	TYPE /FLM/FLANG	FLM: Form Language
IM_FVER	TYPE /FLM/FVER	FLM: Form Version
IM_FID	TYPE /FLM/FID	FLM: Form ID
IM_FID_VAR	TYPE /FLM/ID_VAR	FLM: Form Variant
IM_ACTION	TYPE /FLM/FACTION	FLM: Form Action
VALUE(RE_OWNER)	TYPE XUBNAME	User Name in User Master Record

This method finds the last previous form owner who performed a specific action. It is useful for owner derivation within workflow user-exits for rejection actions.

6.15 Get form name

/FLM/CORE=>GET_FORM_NAME

IM_CCODE	TYPE /FLM/CUST_CODE	FLM: Customer Code
IM_FTYPE	TYPE /FLM/FTYPE_CODE	FLM: Form Type
IM_FLANG	TYPE SPRAS	Language
IM_FVER	TYPE /FLM/FVER	FLM: Form Version
VALUE(EX_FNAME)	TYPE /FLM/FNAME_L	SFS: Long Form Name

This method returns the long name for a form type.

6.16 Get form current owner

/FLM/CORE=>GET_FORM_OWNER

IM_CCODE	TYPE /FLM/CUST_CODE	SFS: Customer Code
IM_FTYPE	TYPE /FLM/FTYPE_CODE	SFS: Form Type
IM_FLANG	TYPE /FLM/FLANG	FLM: Form Language
IM_FVER	TYPE /FLM/FVER	FLM: Form Version
IM_FID	TYPE /FLM/FID	FLM: Form ID
IM_FID_VAR	TYPE /FLM/ID_VAR	FLM: Form Variant
VALUE(RE_OWNER)	TYPE XUBNAME	User Name in User Master Record

This method returns the current owner for a form.

6.17 Get form current status

/FLM/CORE=>GET_FORM_STATUS

IM_CCODE	TYPE /FLM/CUST_CODE	SFS: Customer Code
IM_FTYPE	TYPE /FLM/FTYPE_CODE	SFS: Form Type
IM_FLANG	TYPE /FLM/FLANG	FLM: Form Language
IM_FVER	TYPE /FLM/FVER	FLM: Form Version
IM_FID	TYPE /FLM/FID	FLM: Form ID
IM_FID_VAR	TYPE /FLM/ID_VAR	FLM: Form Variant
VALUE(RE_FSTATUS)	TYPE /FLM/FSTATUS	User Name in User Master Record

This method returns the current status of a form.