FLM 295

Interface Manual

v1.0

4th February 2011

Date	Version	Author	Notes
4/2/11	1.0	Andy Gardner	Initial Version

Table of Contents

1 Overview

This document describes the technical architecture of the FLM solution including a description of the various APIs that can be used to programmatically access the product.

2 Design Time Process

A single FLM Form process consists of the following elements:

- an Adobe Livecycle Designer template (an .xdp file)
- an Adobe Livecycle Designer data schema (an .xsd file)
- an ABAP static class to hold the form process business logic
- a series of entries in FLM client-dependent configuration tables
- one entry in the client-independent configuration table /FLM/FTYPE_CLASS

*2.1 **Adobe Designer Template***

This .xdp file is generated during a run of the FLM Form Wizard and is stored in an operating system directory as specified in the wizard. The file consists of certain graphical elements, a series of scripts and script objects, a series of 'Form Variables' and optionally a data schema file.

This file can be manipulated with the Adobe Livecycle Designer tool, which is installed on the Form Designer desktop as a regular PC application.

During the initial run of the FLM Form Wizard, by default the Adobe Designer Template is taken from a so-called 'Master Template' that is uploaded into the FLM system during initial system setup. This template typically contains graphical elements and styling that are consistent with the customer's overall forms catalogue requirements.

2.1.1 Form Variables

Each Adobe Designer Template has a series of 'Form Variables' embedded inside that help transfer information from the SAP system to the form template at design and run time. These form variables are accessed from the Adobe Livecycle Designer tool using the following menu:

/File

/Form Properties

/Variables

There are several groups of form variables used by the system:

2.1.1.1 FLM Standard Form Variables

The following form variables are present in every form:

Variable	Format	Example	Usage
CCODE	C3	ACL	Customer code
FTYPE	C4	TM01	Form Type*
FVER	N2	00	Form
FLANG	C1	E	Form Language
FID	N10	1000000012	Form ID
FID_VAR	N4	0001	Form Variant
FSTATUS	C2	C1	Form Status
FTRANSPORT	C1	1	Form Transport
REC_EMAIL	String		Receiving email address
RET_EMAIL	String		Return email address
RET_FILE_TYPE	C3	PDF	Return file type
FILLABLE †	C1	X	Interactive or Read only form
FCURRENTUSER†	String	AGARDNER	SAP Account name of logged in user

*Certain form type codes are reserved by the system:

XXXX indicates the Master Template

DOC1 indicates the template used when generating auto-documentation on a form

HTML system internal use for managing HTML forms

CXXX system internal use for managing correspondence forms

JXXX system internal use for managing javascript functions

† These parameters are only available at runtime, not design time.

Customer Code

The customer code is the highest level customising object in FLM and holds various details that affect global system behaviour, such as the name of the authorisation object against which we store FLM user profiles.

Some important points of note about the FLM customer code:

- only 1 customer code can be set to 'default'. Other customer codes may be set up and used for the purposes of prototyping as required. But since at runtime forms can only processed for the default customer, only 1 code can be set as default. In most FLM installations there will only ever be 1 customer code defined.

- customer codes must be unique across all sap clients in the system. FLM is designed run across different SAP clients and we use the customer code and form type to access the ABAP business logic (which is inherently cross-client), hence FLM insists that customer codes maintained manually are unique across all sap clients.

The only exception to this is if you decide to use multiple sap-clients for, for example, development and configuration within the same sap system. In this case, you may migrate the customer definition between clients as required. However, this approach is not recommended as since the form definition and business logic is inherently cross-client, the separation you can achieve is very limited.

Form Type

Form Type code is a 4-character parameter that is the highest point of configuration for a single form process. The code can be freely chosen provided that:

- it begins with a character

- it is not one of the system reserved codes (XXXX, JXXX, CXXX, HTML, DOC1)

The form type code is used throughout FLM to store other configuration details, and is also exposed to the users via the FLM portal.

Form Version

Each form process can have up to 100 different versions indicated by a version number in the range 00 to 99. FLM Versioning is used to separate different versions of the form templates and/or business logic after a form process has been released into production and has gone-live.

To generate a new version, you must execute the FLM Form Wizard and change the version number at the beginning of the wizard. The system will copy the template and schema, and then allow you to make changes to these as required. During upload of the template the system will store the template with version number in the SAP content server.

In order to trigger the system into using the new version, you must write some ABAP code in the form level user-exit 'Version', indicating under what circumstances you wish the new version to be used. This will only cause the system to choose an alternate version for *new* forms; existing forms will retain their existing version number through their entire workflow.

Form Language

The Form Language key indicates to the system which template to use at a particular point in the workflow as determined by the Form Level Language User Exit.

By default only 1 template will be used at every point of the workflow, that of the 'Master Language' as defined in the FLM Customer Code configuration.

For a particular Version of the form process, you may *not* change the data schema between the different languages defined in the system, you may only change the form template itself. That is, every language shares the same data schema at a particular version of the form process.

Form ID

The FLM Form ID is a 10 numeric digit number that is unique to a particular instance of a form process. In a multi-step workflow environment, the Form ID does *not* change between steps (the Form Variant number does).

The Form ID is assigned internally by the system from a standard SAP number range object. The number range object is determined as:

/FLM/XXX	where XXX is your 3 character FLM customer code

The number range Number comes from the FLM Form Types configuration activity in the IMG. During system installation the number range object is created in transaction SNRO and the number range assigned. By default, the number range number '01' is assigned by the FLM Form Wizard, but this can be overridden in the Form Types Configuration IMG activity.

Form Variant

The FLM Form Variant is a 4-digit numeric counter that increments each time a form moves to a new step in the workflow. The first time a form is opened it will have a variant of 0001, the next time 0002 etc. It is this counter and the fact that the system is able to store the form data at each point in the workflow that gives FLM its audit function.

(In older releases of FLM (<FLM 295) it was possible to switch off the audit trail in which case the variant number never exceeded 0001).

Form Status

The FLM Form Status describes the logical position of the form within the workflow (form routing). Typically each point in the workflow has a different status. Statuses have a description and also a 'Status Category' - the Status Category is either:

- Initial Status	(at the beginning of the workflow)
- Intermediate Status	(neither the beginning nor the end of the workflow)
- Final Status	(an endpoint of the workflow)

There can be only one Form Status with a Status Category of 'Initial', but there can be many Form Statuses with a Status Category of 'Final'.

Form Transport

Form Transport describes how the form is sent from the server to the client device. The possibilities are as follows:

1 On Line (default)

2 Off Line as PDF attached to EMail

5 Off Line as Flex Form via Email

6 Off Line as pure EMail for decision capture

Receiving EMail

This is the target email address that the form is to be sent to. It is determined by code written into the Form Level user exit 'EMail'. The eMail is technically sent from the SAP SMTP server to the customer's own EMail server first, it is then sent from there to the receiving party's inbox.

Sending EMail

This is the email address from which the email was sent and is determined from the FLM System Specific Settings table, or a default from the FLM Customer Master record.

Return File Type

The return file type defines how the form is returned to FLM in the offline scenario. You have two choices, either as the full PDF format (which includes attachments, annotations etc) or as just an XML data blob. The XML data version is much smaller than the PDF format, but only includes the pure form data.

You can set the return file type globally on the FLM customer master record, or you can set it individually at the form type level. If the format is set in both places, the form-specific value prevails.

Note that in the standard FLM Master Template this setting only applies if the return transport type is via email (you can configure the return transport type to be via BSP Direct Submit in the IMG activity Form Types Configuration).

Fillable

This is a flag indicating if the form has been rendered as an interactive form ‘value = ‘X’’ or as a read-only form (value = ‘ ‘). This flag helps the javascript developer to adjust the layout of the form in the read-only mode (for example, to remove buttons).

FCurrentUser

This variable holds the SAP logon account name of the user who is currently viewing the form.

2.1.1.2 Form Actions

FLM will create some additional Form Variables which can be used to ensure that the javascript in the form is more readable. Each FLM Form Action has a single Form Variable created for it; below is an example table:

Form Action	Code	Form Variable Name	Value
POST	P	ACTION_POST	P
SUBMIT	S	ACTION_SUBMIT	S

These Form Variables are only available at runtime.

2.1.1.3 Form Statuses

FLM will create some additional Form Variables which can be used to ensure that the javascript in the form is more readable. Each FLM Form Status has a single Form Variable created for it; below is an example table:

Form Status	Code	Form Variable Name	Value
APPROVED	A1	STATUS_POST	A1
REJECTED	R	STATUS_REJECTED	R

These Form Variables are only available at runtime.

2.1.1.4 Colors

FLM will create some additional Form Variables which can be used to ensure that the javascript in the form is more readable. Each FLM Form Color, taken from the IMG Activity called 'Specify Custom Colors' has a single Form Variable created for it; below is an example table:

Color	Form Variable Name	RGB Value
BRIGHT_GREEN	COLOUR_BRIGHT_GREEN	000,200,000

TOPE	COLOUR_TOPE	107,085,067

These Form Variables are only available at runtime.

2.1.2 Adobe Form Data Schema (.xsd File)

The data schema holds a definition of the set of fields on the form together with the set of subforms that the fields map into.

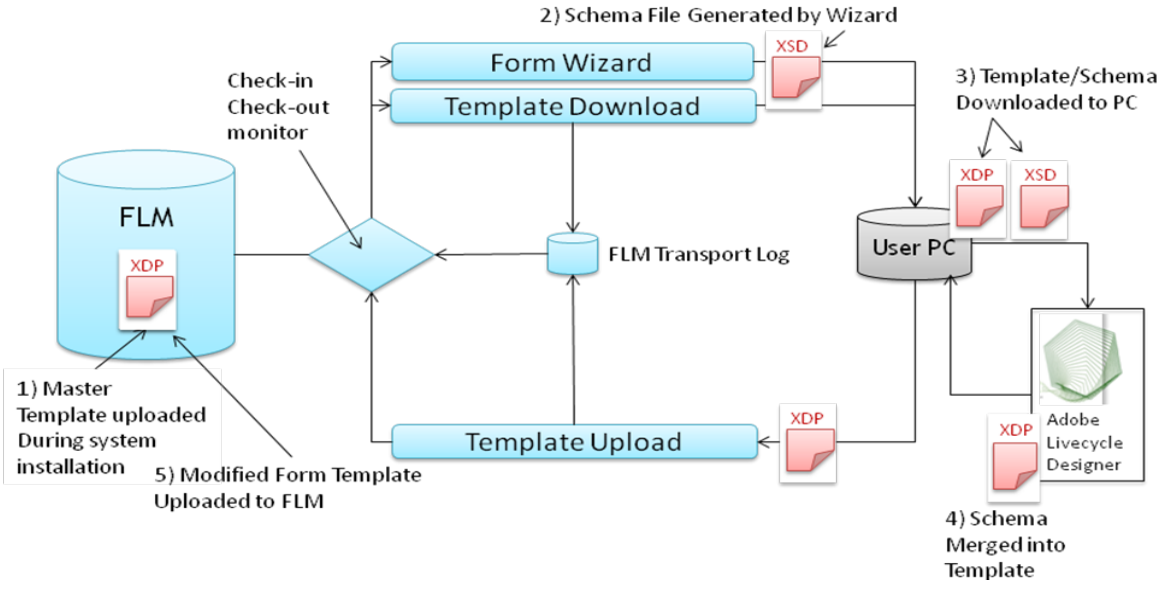
During the execution of the FLM Form Wizard the system will always produce a data schema file on the presentation server in the same way that it produces the Form Template file.

Normally this file is imported into the Adobe Template using a wizard in the Adobe Livecycle Designer tool and once that Adobe Template file has been saved, the Schema is permanently embedded into the Template and hence the two travel around together.

If during subsequent FLM Form Wizard runs the data schema is modified by adding new fields or subforms, the new schema file can be re-imported into the Adobe Livecycle Designer tool. This will overwrite the existing embedded schema without removing any of the existing bindings.

If during subsequent FLM Form Wizard runs the data schema is modified by removing fields or subforms, the new schema file can be re-imported into the Adobe Livecycle Designer tool. This will remove any existing bindings for the fields removed.

The following diagram summarises the lifecycle of the Schema and Template files:



2.2 *Business Logic ABAP Class*

Each form process can optionally have assigned to it some business logic. The implementation of this business logic is contained as ABAP code in a series of automatically generated methods, each of which controls a certain aspect of the form behaviour. We refer to the implementations as 'User Exits', following the traditional SAP model of a User Exit being a point within the code base that the customer can write their own code to control system behaviour.

User Exits and hence business logic occurs at 3 different levels within FLM. Each User Exit has a fixed interface which may contain some data that the system provides (so-called 'import variables') and some data that the User Exit provides back to the system (so-called 'export variables') and some data that does both of those things (so-called 'changing variables').

Each user exit is automatically generated by the system and contains enough documentation to allow the developer to start working with the system immediately. Many user exits also contain sample code.

2.2.1 Customer Level User Exits

User Exits at this level can influence the behaviour of *all* FLM forms within the system. There are 5 different User Exits within the system at this release as follows. Tehnically the ABAP code behind this set of user exits is held in one central FLM class. This class must be migrated manually through the landscape, it is *not* controlled with the FLM Form Transport Wizard.

The linkage between the FLM Customer code and the class name is held on the cross-client table /FLM/FTYPE_CLASS, as an entry under the key XYZ XYZ FLMClassName, where XYZ is the FLM Customer Code.

2.2.1.1 InBox

This user exit allows the developer to control which set of forms are sent to the FLM Portal InBox. By default only forms that are currently owned by the logged in user are sent, but this User Exit allows the developer, for example, to model group access behavior.

The interface definition is as follows:

Import Variables	
im_ccode	FLM Customer
im_user	SAP Account name of Logged in user
Changing Variables	
ch_inbox	This variable contains the standard FLM Inbox as a table, which can be modified and sent back to the system.

2.2.1.2 Server Side Security

This user exit allows the developer to influence the details of certain aspects of the digital certificate that FLM applies to PDFs as they are dispatched from the system.

Import Variables

im_fpe	This is the current record from the /flm/fpe table. This table is the central control table within FLM and holds 1 record for each form at each point in the form routing. This table is described fully under the Central Tables section of this guide.

Changing Variables

ch_type	Type of security to apply. There are 3 possible entries: 1 No-server side security 2 Digitally Sign out-going forms 3 Certify out-going forms
ch_field	Form Field name to apply the signature to (does not apply to certifications)
ch_reason	Human Readable attribute of signature
ch_location	Human Readable attribute of signature designating the location of the Certificate Authority
ch_contact	Human Readable attribute of signature designating the contact name in the Certificate Authority
ch_legalatt	Human Readable attribute of signature designating the legal attestations associated with the certificate
ch_sha_alias	Alias (name) of the actual certificate to apply. The certificates are installed on the java stack according to standard SAP procedures. There are two default certificates “ServerSignature” for digital signatures and “DocumentCertification” for certifications. If it is required to use either of these certificates, this field can be left blank.

Note// certification and digital signing are similar processes. Certification however does not require there to be a physical field on the form, but applies to the entire document. A certified document generates a banner across the top of each page and is therefore easier for the form user to see. Both options render the form ‘tamper evident’, that is, that certain types of changes made subsequently to the form cause the signature to break.

2.2.1.3 PDF Size

This user exit controls the physical size of the FLM Portal window within which the PDF is displayed. It is defined in pixels. In this user exit the developer can, for example, make the PDF size different for different groups of users.

Import Variables

im_ccode	FLM Customer Code
im_ftype	Form Type
im_user	Logged in user

Export Variables

ex_width	Width in pixels (default 1024) in format XXXpx
ex_height	Height in pixels (default 768) in format YYYpx

2.2.1.4 Authorisations

The authorisation model in FLM is fully described in the Authorisations section. In this user exit the developer can override the result of that standard authorisation check. For example to allow access for a user to change only forms from a certain range of customers.

Import Variables

im_ccode	FLM Customer Code
im_ftype	Form Type
im_user	Logged in user
im_fcat	Form Category
im_activity	Activities are as follows:

01	Create form
02	Change form
03	Display form
10	Post Form (via FLM Form Posting Engine)
24	Archive
25	Reload (from Archive)
70	Administer (from FLM Dashboard)

Changing Variables

ch_result	Result of the authorisation check:

0	Check passed
4	Check failed

2.2.1.5 Logon Language

In this user exit the developer can override the results of the standard determination of the users logon language. This language is used to determine the Form Category descriptions, the Form Type Names and various Drop down values in the FLM Portal and also the form template *after* the initial form render (ie for subsequent form users, not for the form creator).

Import Variables

im_ccode	FLM Customer Code
im_user	Logged in user

Changing Variables

ch_lang	Logon language

The standard behaviour of the system is to fill the ch_lang with the default language from the user's SAP master record. If the users are maintained on a remote system, this is also taken into account. If a user does not have a default language maintained, the system master language is used instead.

Note that the template language key is also influenced by the form level user exit 'Language', see below for details.

2.2.2 Form Level User Exits

Form Level user exits influence the behaviour of the system for a particular form type. There are 13 different UserExits at the Form Level. Technically, each set of 13 user exits is held in one FLM ABAP class. This class is migrated through the SAP landscape as part of the FLM Transport Wizard.

The linkage between the form type and the class name is held on the cross-client table /FLM/FTYPE_CLASS under an entry XYZ ABCD FLMFormClass, where XYZ is the FLM Customer Code and ABCD is the FLM Form Type code.

2.2.2.1 Language User Exit

In this UserExit the developer determines which language key is used to retrieve the template during the form *create* process. In subsequent workflow steps (ie form change or display), the language key of the retrieved template comes from the default SAP Logon Language of the subsequent user, which can be influenced by the Customer Level User Exit as described above, if required.

The two ways of determining the template language can be linked together simply by uncommenting the code that is automatically generated in this user exit:

```
*         CALL METHOD /flm/core3=>get_logon_language
*         EXPORTING
*           im_user = im_user
*         RECEIVING
*           re_lang = ex_lang.
```

If this is done, then the template language key is always determined by the Logon Language of the form user.

Import Variables

im_user	Logged in user
im_email_rece	EMail recipient
im_document	External reference document in off-line scenario (char40)
ex_lang	Language key of template to be retrieve

2.2.2.2 Version User Exit

In this user exit the developer controls which version of the form is selected during the form creation process; this version number does not change throughout the process.

Forms have up to 100 versions, in the range 00 to 99.

When a new version is created in the Form Wizard, the old template and data schema are copied into the new version numbers, where they can then be modified as required.

A new version is only necessary after a form process has gone live and a subsequent modification is necessary. In this case the system can manage forms that are in process at the original version, whilst starting new forms at the new

version. It is within this User Exit where the determination of the version number for new forms is done.

Import Variables

im_user	Logged in user
im_email_rece	EMail recipient in offline scenario
im_document	External reference document in off-line scenario (char40)

Export Variable

ex_version	Version key of template to be retrieve

A typical implementation within this UserExit is to determine a new version after a certain system date has elapsed.

2.2.2.3 PrePop User Exit

In this user exit the developer can provide default values to any field on form, and default rows to any table control on the form. Sample code is provided within the User Exit to explain how to perform these operations in different scenarios.

Import Variables

im_user	Logged in user
im_ccode	EMail recipient in offline scenario
im_document	External reference document in off-line scenario
im_ftype	Form Type
im_data	Form Data as internal table type /FLM/XML_TAB_T.

Export Variable

ex_data	Form Data as internal table type /FLM/XML_TAB_T.

The components of the table /FLM/XML_TAB_T are as follows:

NAME	Name of field
VALUE	Value of field
HEIGHT	Height of the node with the form data hierarchy
PARENT	Name of subform containing field
NODETYPE	Always 'Node'
PATH	Describes the position of the field in the XFA hierarchy.

The PATH value is best described by example:

#document/DATA.001/HEADER.001/ITEMS.002/I_COST_CENTRE.001

This example refers to a form field called I_COST_CENTRE. The field sits within the second of a repeating subform called ITEMS which in turn sits within a subform called HEADER. The leading string "#document/DATA.001" is static.

Within this User Exit the developer can modify the contents of the internal table ex_data to populate data into the form.

By default this UserExit only fires when a form is created, or during a check-cycle, and never again. However, if it is required for this to fire at subsequent points in the workflow it can be configured to fire at different statuses in the IMG activity 'Set Additional PrePopulation Statuses'.

2.2.2.4 Template User Exit

In this User Exit the developer can change the template before it is sent to ADS for rendering.

Import Variables

im_fpe	Current /FLM/FPE record
im_user	Logged in User
im_annotations	Form Annotations
im_attachments	Form Attachments
im_fstatus	Current form Status
im_document	External reference document
im_data	Table of Form data
im_fillable	Flag indicating if form is fillable ('X') or read-only (' ')

Changing Variables

ch_xdp	Form Template

This User Exit fires immediately before the template is sent to ADS for rendering and allows the developer to modify the template by, for example, inserting an image file dynamically into the template.

2.2.2.5 Routing User Exit

In this User Exit the developer controls the routing (workflow) that the form follows.

Import Variables

im_instance	Current record from table /FLM/FPE
im_status	Proposed new form status from routing status table
im_action	Form action chosen by the previous user
im_notif	Proposed Notification flag from configuration
im_ftransport	Proposed Transport Type
im_sig	Details of digital signatures held within form
im_atts	Current form attachments

Export Variables

ex_owner	New form owner
ex_notif	New Notification flag ('-' => No Notification required 'x'=>Notification required ' '=>Use FLM configuration
ex_status	New form status
ex_ftransport	New form transport
ex_task	New Portal Task Instructions
ex_notif_title_text	Text of Notification Email Title
ex_notif_body_text	Text of Notification Email body

In addition the form data is available to the developer by uncommenting the code:

```
*  DATA: l_form_data TYPE /flm/xml_tab_t.
*
*  CALL METHOD /flm/core=>get_data_from_instance
*    EXPORTING
*      im_form_instance = im_instance
*    RECEIVING
*      ex_form_data     = l_form_data.
```

Typically in this user exit the developer is determining the next form owner and the next form status.

2.2.2.6 Index User Exit

In this UserExit the developer writes custom data to the FLM Indexing fields. The contents of these fields can either be determined by standard configuration (IMG Activity 'Form Types Configuration') or via this User Exit. If the FLM standard delivered index fields have been extended via FLM Note 11, then the application

developer has no choice but to use this UserExit to provide values into the custom structure.

Import Variables

im_data	Form Data
im_ccode	Customer code
im_ftype	Form Type code
im_document	External reference document
im_user	Logged in User

Export Variables

ex_findex	Fields from structure /FLM/FINDEX

Typical usage is for when an index field must be calculated from other form data, rather than simply being transferred from a single form field.

2.2.2.7 PDF User Exit

This User Exit allows the developer to access the PDF after it has been rendered by ADS. Typically this would involve sending the PDF to an external archive system, or to have the PDF digitally signed by an external system.

This user-exit is called:

1) after ADS has rendered the pdf and before the pdf is sent to the user via any channel. In this case you can use this userexit to modify the pdf, for example to add a signature or password.

2) after the user has submitted the form (via any channel). In this case the parameter im_faction will have a value. You cannot update the pdf in this scenario, but can for example archive or copy the pdf.

Import Variables

im_ccode	Customer Code
im_ftype	Form Type Code
im_fstatus	Form status
im_faction	Form action
im_fillable	Form fillable flag
im_data_string	Form data as xstring
im_template_string	Form template as xstring

im_attachments	Form attachments
im_annotations	Form annotations
im_cms	Content Management System reference of form (only available as form is inbound to SAP)

Export Variables

Messages can be sent back in parameter ex_mess, these will be logged in the system log and under online scenarios sent back to the user.

Changing Variables

ch_pdf	PDF changing parameter.

2.2.2.8 e-mail User Exit

In this User Exit the developer determines the e-mail address of the receiver of the form.

This user-exit is used for 2 tasks:

1) Generating e-mail address(es) at the start of a process for a mass-mail shot type process. In this case you may generate more than one email address and also use the im_document parameter.

2) Generating a single email address for a form routing that is going off-line at a certain status. In this case, only the first email generated will be used, there is no application document reference, but you can also use the additional form data. Note, this user exit is called during inbound processing after the routing user-exit has been called.

If you set the flag HTML in the export parameter `ex_email_addrs`, the email will be formatted as an HTML-type email.

Import Variables

im_email_stat	Form status
im_document	External reference document number
im_fpe	Record from table /FLM/FPE
im_wf_stat	New routing status
im_owner	New form owner after routing

Export Variable

ex_email_addrs	Table of email addresses.

2.2.2.9 Enqueue User Exit

This User Exit is used for locking an object in SAP or FLM as a form is dispatched out of the system. Typically this is only useful when a form is being dispatched via e-mail.

Import Variables

im_data	Form data
im_ccode	Customer Code
im_ftype	Form Type Code
im_document	Application document number
im_user	Logged in user

Export Variable

ex_subrc	Return code of locking operation. If you set this flag to non-zero you inform FLM of a failure situation and the system will respond with an error message "Object lock entry could not be written" in the FLM system log. Processing of the form also halts.

A typical usage is to, for example, lock the details of a customer master record whilst a change of details form is out with the customer.

2.2.2.10 Dequeue User Exit

This is the opposite user exit to the Enqueue User Exit above in that it unlocks an object previously locked.

This is the only FLM User Exit that is not called automatically by the system at some point in the processing cycle. Rather, it is designed to be called from within a posting adaptor via the FLM Form Posting Engine (FPE).

Import Variables

im_data	Form data
im_ccode	Customer Code
im_ftype	Form Type Code
im_document	Application document number

Export Variable

Update return code in the Variable ex_subrc.

2.2.2.11 Factory Cal User Exit

This user-exit is called during the execution of report /FLM/WF_ENGINE, a scheduled report used for determining escalations and reminders. The Factory Calendar is used by the system to determine the calendar date for form reminders and escalations and can be configured globally (against the FLM customer code) or

in a form-specific configuration table. This User Exit allows the application developer to determine the factory calendar against more complex criteria.

Import Parameter

im_fpe	A record from the table /FLM/FPE

Changing Variable

ch_fcal	Proposed Factory calendar from FLM configuration

2.2.2.12 FPE Alert User Exit

This User Exit is called from the Forms Posting Engine in the event of a posting adaptor failing, with the intent of formatting and sending an email alert to the system administrator.

Import Variables

im_fpe	A record from the table /FLM/FPE
im_fpe_status_t	FPE Status table of type /FLM/FPE_STATUS_T

Export Variable

ex_wf_remi	New /FLM/WF_REMI record containing new target email address and 2 texts:- email address - ex_wf_remi-err_receiver email title text - ex_wf_remi-err_title email body text - ex_wf_remi-err_body

By implementing this User Exit, each form type can have a different FPE system administrator.

2.2.2.13 Notification/Reminders User Exit

This user-exit is called in two places:

1) As part of the scheduled report /FLM/WF_ENGINE generating reminders and escalations.
2) During routing for those forms that have a notification step

Import Variables

im_notif	'X' indicates notification processing ' ' indicates reminder processing
im_fpe	FPE Record. Form owner is in im_fpe-fowner.
im_user	Standard FLM-determined receiving user

Changing Variable

ch_email_addrs	List of email addresses to send reminders /

notifications to. The first line of this table is prefilled with the email address associated with im_user above.

2.2.3 Field Level User Exits

There are 5 different User Exit types that operate at the field level. To be clear, this means that we generate a code block that is only executed for a particular field on a particular form to perform a particular function.

The main technical difference between the Field level UEs and the Customer and Form Level UEs is that the Field Level UEs are *not* automatically generated; during the design of the data schema in the FLM Form Wizard, the form developer nominates which fields are to have which UEs associated with them. This is done for performance reasons.

The 5 different types of User Exit are described below.

2.2.3.1 DropDown List Box Field Level User Exit (F4 Entries)

This User Exit is concerned with the determination of a set of possible entries to be inserted into a drop-down list box UI element on the form.

Import Variables

im_data	Form data
im_ccode	Customer Code
im_ftype	Form Type Code
im_field	Current field name
im_doc	Application document number
im_user	Logged in user

Export Variable

ex_form_data	Internal table of possible entries, table type /FLM/SFS_FORM_DATA_T

The table is a set of name, value pairs, where:

NAME	Code
VALUE	Human readable description for drop down

User selected values are have their *NAME* value placed into the corresponding node in the data schema once the form is submitted back to FLM.

2.2.3.2 Pre Population Field Level User Exit

This User Exit allows the developer to provide a single field with a default value. Unlike the Form Level User Exit, this UE cannot be used to provide default rows into repeating subforms, and can only be used to provide a default value for the single field for which it is generated.

Import Variables

im_data	Form data
im_ccode	Customer Code
im_ftype	Form Type Code
im_field	Current field name
im_value	Current field value
im_document	Application document number
im_user	Logged in user

Export Variable

ex_value	Default Value

Similarly to the Form Level Prepopulation User Exit, by default this User Exit is only fired during the Form Creation process and not during subsequent Change Form steps. Again, if it is required for this UE to fire at other statuses, it can be configured to do so via the FLM IMG activity 'Set Additional PrePopulation Statuses'.

2.2.3.3 Validation Field Level User Exit

This is a dual-use User Exit – firstly it allows the developer to define certain rules that fire after the form is submitted; in the event of the rule failing, the form is returned to the user's screen with an error message as defined in this User Exit.

Secondly, it allows for the form attachments or annotations to be changed.

Import Variables

im_data	Form data
im_ccode	Customer Code
im_ftype	Form Type Code
im_field	Current field name
im_value	Current field value
im_path	Path of current field
im_return	Form Action + CMS Doc + Email Address

Changing Variables

ch_atts	Attachments
ch_anns	Annotations

Export Variables

ex_response String composed of up to one On-line and one Off-line code:

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 customer message class
ex_msgvar1 Error variable 1
ex_msgvar2 Error variable 2
ex_msgvar3 Error variable 3
ex_msgvar4 Error variable 4

The customer message class is constructed from the FLM Customer Code as follows: /FLM/XYZ where XYZ is the FLM Customer Code.

2.2.3.4 Substitution Field Level User Exit

This User Exit is used for replacing the value a user has entered onto the form with an alternate value for business reasons.

Import Variables

im_data Form data
im_ccode Customer Code
im_ftype Form Type Code
im_field Current field name
im_value Current field value
im_path Path of current field
im_return Form Action + CMS Doc + Email Address

Export Variable

ex_value New value

2.2.3.5 Derivation Field Level User Exit

This User Exit allows the developer to derive a value for a field that is not determined by the user entering a value onto the form. In this way it is possible to construct other values within the data schema as required.

Import Variables

im_data Form data
im_ccode Customer Code

im_ftype	Form Type Code
im_field	Current field name
im_value	Current field value
im_path	Path of current field
im_return	Form Action + CMS Doc + Email Address

Export Variable

ex_value	New value