

006

GUIDELINES ON MODULE CREATION

The copying, distribution and utilization of this documents as well as the communication of its contents to others without expressed authorization is prohibited.
Offenders will be held liable for the payment of damages. All rights reserved in the event of the grant of patent, utility model or ornamental design registration.

Document Signature Table

Name	Function Company	Signature	Date
<i>Written by:</i>	Mame Coumba Sall Sébastien Robin Thierry Brettnacher Jérôme Perrin		
<i>Approved by:</i>			
<i>Released by:</i>			
<i>Accepted by:</i>			

Distribution List

© Nexedi SARL / ISO 16016

Table of Contents

1 Introduction.....	7
1.1 Purpose and Scope.....	7
1.2 References.....	7
1.2.1 Applicable Documents.....	7
1.2.2 Referenced Documents.....	7
1.3 Abbreviations and Definitions.....	8
2 Rules.....	9
2.1 General Rules.....	9
2.2 Portal Types.....	9
2.2.1 Module Portal Types.....	9
2.2.2 Document Portal Types	12
2.3 Forms.....	14
2.3.1 General Rules.....	14
2.3.2 Listbox Rules.....	18
2.3.3 Module forms (List Mode)	18
2.3.4 Document Forms (Document view mode).....	20
2.4 Activities.....	22
2.5 Workflows.....	22
2.6 Security.....	26
2.6.1 Modules.....	26
2.6.2 Documents.....	27
3 What are different action categories and how are they used on the front page.....	28
3.1 Quick Search	28
3.2 Contribute	28
3.3 Browse	29
3.4 New	29
3.5 Dig	29
3.6 Reports	29
3.7 Printouts	30
3.8 Exchange	31
3.9 Express Support	31
4 How to use proxify tool and migrate existing business templates to proxyfields.....	32
1. Think how to group fields in field libraries.....	32

2. Create conceptual field libraries.....	32
3. Use field libraries.....	33
4. Check the result.....	33

1 Introduction

1.1 Purpose and Scope

The purpose of this document is to define a set of rules to follow when creating a new module.

This document covers organisational approach to software development. It does not contain any information to improve technical skills of developers.

This document is meant to be read by Nexedi developers and by selected Nexedi partners. This document must not be distributed outside Nexedi without prior explicit permission of Nexedi CEO.

1.2 References

The section provides the lists of the applicable and reference.

1.2.1 *Applicable Documents*

ID	Title / Reference	Doc. No.	Issue/ Rev.
AD-1		DP	N/A

1.2.2 *Referenced Documents*

ID	Title / Reference	Doc. No.	Issue/ Rev.
RD-1	ERP5 Implementation Process	ERP5-IMPL	Issue 1.0 Rev. 0

ID	Title / Reference	Doc. No.	Issue/ Rev.
RD-2	Guidelines for Naming Convention		

1.3 Abbreviations and Definitions

The section provides the lists of the applicable abbreviations.

ID	Abbrevi- ation	Title	Description
AB-1			
AB-2			
AB-3			
AB-4			
AB-5			
AB-6			
AB-7			
AB-8			

2 Rules

This chapter describes the different rules to follow in order to create a module.

2.1 General Rules

Rule: Always follow the erp5 naming convention defined here : <http://www.erp5.org/GuidelinesForNaming-Convention>

Rationale: All ERP5 developers must follow the ERP5 naming convention which is a list of naming convention guidelines for portal types names, module titles, script titles, form titles, field titles, etc.

All script names, portal_types names, form names, field names, workflow state names, etc. should be in “British English” and are translated in other languages with the Localizer tool.

2.2 Portal Types

2.2.1 Module Portal Types

Rule: Define portal type ids according to the erp5 naming convention

Rationale: The portal types names should follow the erp5 naming convention

All portal_types IDs should be defined in “British English” and are translated in other languages with the Localizer tool.

Rule: Don't define portal type title

Rationale: The portal type titles are not used.

Portal Types titles are not used, it's useless to put the same thing as in ID, and it's a mistake to have something different. So it's better to leave them empty. If a portal type define a title, the title have to be removed.

Rule: Give a brief description of your module in the Properties Tab of the portal type

Rationale: The description is important to explain what the module consists of.

You should for example describe what are the documents contained in the module.

Rule: Configure title translation to use `erp5_ui` domain

Rationale: Modules titles are part of the UI and shall be translated with `erp5_ui`

`getTranslatedTitle` method will use Localizer's *erp5_ui* catalog to translate the title of the module document when displayed in the page templates.

Rule: In the Actions Tab, define the actions available on the module

Rationale: Only the actions that are really needed on the module should be present. The minimal list of actions is provided below.

The minimal list of actions on a module is : **View**. Those actions should be defined as follows :

- View : permission *View*, category *object_list* and id *view* and action string: `{object_url}/ModulePortalType_viewDocumentPortalTypeList`

Other default actions (Sort, Search, Print, Modify UI, Import, Export and Change State) are defined as *global actions* on `portal_actions` tool, you don't have to include them as portal type actions (like it was done before). Those actions are now replaced global actions, if they are already present on the portal type, you have to remove them :

- Sort : permission *View*, category *object_sort*, id *sort_on* and action string: `${object_url}/Folder_viewSortOnDialog`
- Search : permission *View*, category *object_search*, id *search* and action string: `${object_url}/Folder_viewSearchDialog`
- Modify UI : permission *View*, category *object_ui*, id *list_ui* and action string: `${object_url}/Base_viewUIDialog`
- Print : permission *View*, category *object_print*, id *print* and action string: `${object_url}/Base_printPdf`
- Import Csv File : permission *Modify portal content*, category *object_exchange*, id *csv_import* and action string: `${object_url}/Base_importCsvFileForm`
- Export Csv File : permission *View*, category *object_exchange*, id *csv_export* and action string: `${object_url}/Base_exportCsvFileForm`
- Change Workflow State : permission *Add portal content*, category *object_workflow*, id *change_workflow_state* and action string: `${object_url}/Base_changeWorkflowState`

Rule: If you add actions to the minimal set of actions on a document, their names should be in “British English”. These names should be explicit compared to the action

Rationale: The names are displayed to the user in the action menu on the document and is used for translation in other languages

Actions names should always be defined from an use case and end user point of view and not from a technical point of view.

Rule: If you define “report” actions, they have to be associated to “View” permission and have to be categorised as *object_report* action.

Rationale: A specific fast report button is provided inside the action bar of ERP5 when needed

A well configured ERP5 site provides reports for almost all modules.

2.2.2 Document Portal Types

Rule: Define portal type ids according to the erp5 naming convention

Rationale: The portal types names should follow the erp5 naming convention

All portal_types IDs should be defined in “British English” and are translated in other languages with the *Localizer* tool.

Rule: Don't define portal type title

Rationale: The portal type titles are not used.

Portal Types titles are not used, it's useless to put the same thing as in ID, and it's a mistake to have something different. So it's better to leave them empty. If a portal type define a title, the title have to be removed.

Rule: Give a brief description of your document type in the Properties Tab of the portal type

Rationale: The description is important to explain what is the purpose of this type of document.

You should for example describe what are those documents used for.

Rule: Configure title translation to use no translation domain

Rationale: For most cases, document's title does not have to be translated.

Sometimes in a multilingual environment you may want to translate document's title, in that case you should use erp5_content. Keep in mind that this may add lots of messages in you Message catalog, and cause database size problems.

Rule: If you plan to use “fast inputs” to create content types inside the portal type, you have to define these content types as “hidden content types” in the Properties Tab of the portal type.

Rationale: If the Use Case Analysis leads to the need of automatic generation methods for content documents using fast inputs, then manual creation of these content documents should not be proposed to the user

You should refer to Use cases to know if fast inputs are needed or not.

Rule: In the Actions Tab, define the actions available on the document

Rationale: Only the actions that are really needed on the document should be present. The minimal list of actions and their order is provided below.

The minimal list of actions on a document is : View. Those actions should be defined as follows :

- View : permission *View*, category *object_view* and id *view* and action `string:${object_url}/PortalType_view`

Other actions are defined as global actions, they will automatically be present. If actions already exists on the portal type, you should remove them. Actions added for a document are:

- Print : permission *View*, category *object_print*, id *print* and action `string:${object_url}/Base_printPdf`
- History : permission *View*, category *object_view*, id *history* and action `string:${object_url}/Base_viewHistory`
- Metadata : permission *Manage properties*, category *object_view*, id *metadata* and action `string:${object_url}/Base_viewMetadata`

The order of the actions of category *object_view* (which are defining tabs in the user interface) has to be the following : view, other views (example : details), history, metadata.

For the ZMI view, the order goes from top to bottom. You should define the priority of actions according to that order (this is also valid for the modules portal types).

Rule: If you add actions to the minimal set of actions on a document, their names should be named in "British English". These names should be explicit compared to the action

Rationale: The actions names are displayed to the user in the action menu on the document and is used for translation in other languages

Actions names should always be defined from an use case and end user point of view and not from a technical point of view.

Rule: If you define "fast input" actions, they usually have to be associated to "Add portal content" permission and have to be categorised as `object_fast_input`.

Rationale: A specific fast input button is provided inside the action bar of ERP5 when needed

Never configure a fast input action as *object_exchange* action category, use *object_fast_input* action category.

In some cases, if fast inputs are used to modify existing document, the fast input action should rather be associated to *Modify portal content* permission, if they create new sub-documents, the action should use *Add portal content* permission.

2.3 Forms

2.3.1 General Rules

Rule: Use appropriate titles in "British English" for the form and the fields titles so that localisation becomes easy. Titles of forms and fields have to be chosen carefully and must make sense from a business point of view.

Rationale: The user interface has to be "business oriented" and localisation work of the user interface has to be easy

ERP5 technical vocabulary should be banished from the user interface.

Example 1 : "Shipping Date" is preferred to "Start Date" on a packing list.

Example 2 : "Customer" is preferred to "Destination Section" on a sale order.

Rule: All fields must have a correct description.

Rationale: Description is used to generate a tool tip.

This description can be on the property sheet, or on the field. If the description of the property sheet definition is not good on the context (property sheet descriptions often are too technical), the description must be set on fields (on the first level of proxyfield).

Rule: The Id of an ERP5 document should never be handled by users or shown to them in View or List Forms

Rationale: Id is a technical property in ERP5 and should not be significant for users.

Ids should never (and no longer) be used in ERP5 Forms. For an ERP5 user, an ERP5 document has to be identified by either its title, reference or int_index but not by its id which is a technical property.

Rule: Use proxyfields of proxyfields to create forms. Each Business Template provides libraries of fields that have to be used to define the forms associated to modules and documents.

Rationale: All the forms of an ERP5 site have to be consistent. The maintenance of the forms has to be easy.

The fields used to define forms are proxyfields of proxyfields. Actually ERP5 Base provides a field library in which the default rendering of each kind of field is defined.

The field libraries contained in other business templates are defining the logic associated to the field.

Example : Shipping Date

In the Base field library, a “Date” field is defining that dates are displayed using the date format chosen by the user in its preference document.

In the Trade BT, an order field library defines that “Shipping Date” is a proxy-field of the “Date ” field and that it will handle the start_date property.

In the order or packing_list view form, a proxyfield of “Shipping Date” is used to display the shipping date.

Rule: Avoid check required for a field in a view form

Rationale: This kind of constraint should be defined in Constraints documents and have to be checked in workflow transitions.

Checking a field as “required” can cause problems like not saving the data when the user clicks on the save button. The same apply for the use of *External Validators* scripts in fields.

You should not use it unless it cannot be checked using constraints or can put

the system in an unstable state. For instance, web page titles, documents references are required fields because otherwise we couldn't find those documents. Password fields on the Person view have a confirmation with an external validator, because we don't want to lock out an user that mistakes his password.

Notice that it is accepted to define a field as required in dialogs.

Thinking that a property will always be entered by the user is a mistake, you may think that a person will always have a name, but in real life for example you will find some cases like: "Mr Smith" without first name, "Betty" from Supplier company etc..

Rule: If a form contains a listbox with editable fields by `listbox_property`, those fields should be made hidden in the form by moving them in the "hidden" group

Rationale: Fields in hidden groups are not rendered at all (this is different from checking "hidden" property on field, in this case it's rendered as a `<input type='hidden'>`)

If the hidden group does not exist in the form, you should create it

Rule: All the fields representing `simulation_state` and `validation_state` should be `my_translated_simulation_state_title` and `my_translated_validation_state_title`

Rationale: The workflow states should be automatically translated when the user changes the current language.

The workflow states titles are translated using Localizer.

Rule: In view mode, `simulation_state` or `validation_state` fields should be in the bottom of "right" group

Rationale: Forms must be consistent.

Rule: Always use the "item list method" from when you configure a `ListField` or `MultiListField` to handle categories in forms.

Rationale: The way to render categories in pop-up menus must be consistent across various forms and must be configured from user preferences.

The categories titles are translated using Localizer (`erp5_content`).

The categories must be sorted by `int_index`, then by translated title, while respecting the hierarchical structure. The idea behind this is to have for example this ordering: A, B, P, Z, Others. With “Others” in last position, to achieve this, we have to set and `int_index` on “Others” category with a value greater than 0 (which is the default for A, B, P, Z).

The categories that cannot be viewed (ie. no *View* permission) by the user should not be displayed.

The method must be defined on the first level field library, and use the value *preferred_category_child_item_list_method_id* from preferences. The “items” TALES for the first field is horrible, but generic:

```
python: getattr(here.portal_categories[field.getId().replace('my_', '', 1)], preferences.getPreference('preferred_category_child_item_list_method_id', 'getCategoryChildCompactLogicalPathItemList'))
(local_sort_id=('int_index', 'translated_title'), checked_permission='View')
```

In some cases, the method used to render a list of categories could be different. For example, when handling values of the function `category` on a person, only leaves of the category tree should be displayed.

Rule: Use always `translated_portal_type` if you want to display the portal type of a document in a form.

Rationale: This allows to provide automatic translation of the `portal_type` property.

The portal types titles are translated using *Localizer* (`erp5_ui`).

Rule: An ERP5 form in view mode usually contains 5 groups named : left, right, center, bottom and hidden. Exception can be made for portal types with lots of properties, like Business Template.

Rationale: These groups are needed to do a proper rendering of the form

Only the “bottom” and “hidden” groups are used in list mode. That means that “bottom” and “hidden” are the only groups required when you are designing a form to render the content of a module. For view mode, you need at least left, right and center; and bottom if your view contains a listbox.

The groups are defined in the 'Order' tab of the form.

Rule: The information can be splitted in multiple tabs

Rationale: When there is too much information to fit on one single form, multiple views can be shown.

If using multiple tabs, the first tab should contain the most useful information.

Information from the same concept should be on the same tab, for example on an order you have a tab for general information, payment information or discount information.

The same information should not be present on two different views, one exception is the profile tab that is on all deliveries types.

If a view only contains a listbox, a read only "title" field should be shown in the "center" group.

2.3.2 Listbox Rules

Rule: Never check the "Search Row" and "Select Column" properties for a listbox included in a document view form. These properties are reserved for listboxes used to view a module.

Rationale: Searching and selecting documents in a listbox are not allowed in view mode

It is allowed to display several listboxes (without "search row" et "select column") inside a single view form.

Rule: Always display the translated_portal_type of documents in the listbox of a module if this module can contain different portal types

Rationale: If a listbox in ERP5 is supposed to display objects having different portal types, then a portal type column is needed in the listbox to show this information. What should be displayed is always the "translated_portal_type" so that the forms can be automatically localised.

If a listbox is configured so that only objects from one portal type can be displayed, then the title of the listbox should be set to the plural of the given portal type title (ex : Sale Orders should be the title of a listbox displaying "Sale Order" objects).

2.3.3 Module forms (List Mode)

Rule: The id of the form should be as follows: *ModulePortalType_viewPortalTypeList*. The form title should always represent what is displayed.

Rationale: This is defined in ERP5 naming convention

The page template used is 'form_list'

The form method is 'POST'

The form action is Base_doSelect

The encoding is UTF-8

Rule: Module form should only contain a listbox (and possibly editable fields) with id "listbox", in group "bottom"

Rationale: Default view for modules is always a listbox in list mode, its ID is by convention "listbox".

Rule: Define in the 'More Column' fields all the properties that an user will be able to display for a main listbox of a module

Rationale: The user has to be able to select which columns he wants to see in order to view the content of a module

The more columns field is also used in order to generate a default search dialog on a module.

This field has to be defined only for listboxes used in list mode (not used for listboxes inside view forms).

Include all properties that can be useful, like creation date, modification date, creator ...

Rule: When displaying a date in a listbox, always configure a specific listbox_date editable Date-Time field in the form

Rationale: This allows that the date is displayed in the listbox according to the user date format preference

In ERP5, the most used searchable dates are delivery.start_date and delivery.stop_date. Using these dates in a listbox allows to do some filtering by dates. The column must be named "delivery.start_date", and the editable field must have id "delivery_start_date"

Rule: By default, the name of the selection inside the listbox should be set follow "module_name_selection" naming (module_name will be the id of the module in lowercase)

Rationale: The selection_name is used to find in a dictionary the parameters used inside a listbox. Generally we want a different selection_name for each listbox.

In some cases, you could set the selection_name to another value.

Rule: "List Method" should usually be defined as "searchFolder" for listboxes used to view modules.

Rationale: searchFolder allows to show the content of a folder (module) efficiently by using the ERP5 SQL Catalog.

The method used to display documents in a module must use SQL for scalability.

Rule: "Count Method" should usually be defined as "countFolder" for listboxes used to view modules.

Rationale: countFolder allows to show the number of document in a folder efficiently by using the ERP5 SQL Catalog. This result is also used to limit the results to retrieve using the list method.

The method used to count documents in a module must use SQL for scalability.

2.3.4 Document Forms (Document view mode)

Rule: The id of the form should be as follows: *PortalType_view*. The form title should always represent what is displayed. Generally the title of the form is the portal type title.

Rationale: This is defined in ERP5 naming convention

The page template used is 'form_view'

The form method is 'POST'

The form enctype is multipart/form-data (actually only important if you have file uploads)

The form action is Base_edit

The encoding is UTF-8

Rule: Do not mix up description and comments

Rationale: The meaning of those 2 properties is different

« Description » should be used to enter a short (one or two sentences) description of a document. « Comments » should be used to enter miscellaneous information concerning the document. All the information that can not take place in any other field or property should be considered as « comment ».

Comments should take place in the « right » group of a form as a small text area field. Description should take place in the « center » group as a large text area field.

Rule: Relation string fields should never permit to define relations with categories objects (portal type Category)

Rationale: The notion of category is not easy to understand for an end user and having to select a correct category value among a huge list is bad interface.

Category should not be present in the Portal Type field of a relation string field (example : my_source_title, my_destination_title, my_resource_title field)

Rule: If the form contains a listbox, the corresponding selection should be named as follows : `form_name_selection_name` (`form_name` is the id of the form in lowercase)

Rationale: The `selection_name` is used to find in a dictionary the parameters used inside a listbox. Generally we want a different `selection_name` for each listbox.

In some cases, you could set the `selection_name` to another value.

Rule: If the form contains a listbox, the corresponding "List Method" will be most of the time `contentValues`.

Rationale: Generally we want to display sub documents

If you are not displaying sub objects but related objects, you will need to define another List Method, often category accessor, like `getSourceProjectRelatedValueList`.

In "default parameters" you have to specify "`checked_permission | View`" so that `contentValues` does not display sub-documents on which the user does not have the *View* permission.

`searchFolder` method uses SQL catalog to get documents so if the number of subdocuments is known to be small, it's usually faster to use `contentValues` directly; `contentValues` gets documents from Zope's object database.

Rule: If the form contains a listbox, the corresponding "List Action" should be set to the id of a form displaying the same listbox but in List mode.

Rationale: When clicking on the title of a listbox, we want the same listbox to be displayed in List mode so that selecting, deleting, filtering, etc. features become available.

The form associated to the List action of a listbox should just contain a proxy-field of the initial listbox.

Rule: Use `getTranslatedId` when you want to display the special ids of a document in a listbox or in a form (ex : `default_address` sub object)

Rationale: Objects with reserved ids like "`default_address`" have a special meaning and behaviour in ERP5. It is necessary to display this information to the user and to be able to do some localisation.

XXX open questions:

How to always have a link ?

2.4 Activities

Rule: Use activities to do heavy calculations in the background

Rationale: To have a responsive system, CMFActivity system should be used to defer execution heavy tasks.

Indexing and simulation are done as activities.

Rule: Set priorities on non urgent activities so that important activities like indexing are executed first

Rationale: If your activities are not urgent, using a priority of 2 or 3 gives a higher probability that more important activities will be executed first

2.5 Workflows

Rule: Every portal type used as first level document in a module should be associated to a "validation" workflow.

Rationale: The life cycle of all main documents is defined by a workflow in ERP5.

Security in ERP5 is also workflow based.

Rule: The workflow title should be defined and translated.

Rationale: It is needed to display workflow history on a document

The workflow titles are translated using Localizer (erp5_ui).

Rule: Always prefer `simulation_state` or `validation_state` for “workflow state variable”

Rationale: `simulation_state` and `validation_state` can be searched using `portal_catalog`.

It's preferred to use `validation_state` for documents with a validation cycle and `simulation_state` for movements and deliveries.

Note that it's also important that 2 workflows with the same state variable are not used for one document type. If you have to workflows with `simulation_state`, document accessors like `getSimulationState` will not be able to determine which workflow actually sets the `simulation_state`.

Rule: All “validation” workflows must define “Deleted” state and “delete_action” transition.

Rationale: In ERP5, deleting a document is part of the life cycle of the document

Security should be configured so that only managers can view documents in “Deleted” state. That means, for a common end user, deletion of documents in ERP5 should be seen as “physical” deletion.

The “delete_action” must be configured as follow: “Trigger type” must be set to “Initiated by user action”, “Name (formatted)” must remain empty, so that this workflow action is not shown in view mode actions. You can specify a guard if needed.

As usual, `delete_action` transition should not have a destination state (“remain in state”) and have the “delete” transition as “script after”. This “delete” action will have “Trigger Type” “Initiated by WorkflowMethod” and destination state “deleted”.

Rule: Define the permissions used by the workflow to 'View' (to view the object), 'Modify portal content' (to modify all information on the object) 'Access Content Information' (to access/ to view all data saved on the object), 'Add Portal Content' (add actions on the document) and 'Delete objects' (delete sub-objects).

Rationale: This allows you to redefine those permissions for each workflow states of the document.

Refer to ERP5 Security Guidelines in order to learn more about security.

Rule: Define the state titles so that it makes sense from a business point of view

Rationale: The user must understand immediately the meaning of a workflow state and localisation should be easy.

For example, according to the standard movements management in ERP5, for packing lists there is a workflow state named “started” witch has the sense of “shipped”. The Id of this state should be “started” and its title should be “Shipped”.

Rule: Define the transition titles so that it makes sense from a business point of view

Rationale: The user must understand immediately the meaning of a workflow transition and localisation should be easy.

For example, the “start_action” transition for a packing list has the sense of “ship”. So the corresponding title should be “Ship”.

You should also define a name for the “Display in actions box”. This name corresponds to what will be displayed to the user in the list of actions.

The title field is used for the history display of the document.

Rule: All the scripts that should validate the workflow transitions should be well defined

Rationale: It is very important to do checks during the validation step to ensure that all data is consistent and that everything is well defined.

Checking the consistency of the data in ERP5 is done on user action workflow transitions and not when entering data.

“Permanent” checks have to be implemented in constraints. Permanent checks are checks that should apply at any time, for example that an order has a source and a destination. The workflow script only have to call `sci['object'].Base_checkConsistency()` to perform constraint based checks.

“Temporary” checks have to be implemented in the workflow script. Temporary checks are checks that should only apply when we try to do the action, for example that client information is valid at the time we send an invoice.

Rule: Validation scripts must return translated messages

Rationale: Messages from validation scripts will be present in the application's status message and in the workflow history

Use `Products.ERP5Type.Message` class so that translation is done only when the message is displayed to the user (this allow translation of workflow history).

Rule: Define the roles allowed to execute the transitions

Rationale: This is required to be compatible with ERP5 5A security model

Refer to security and use cases specifications in order to know witch roles should be associated to a given transition.

Rule: Always prefer using the default dialog (`Base_viewWorkflowActionDialog`) for transitions

Rationale: This form allow to pass a workflow transition and specify a comment. If you need to pass more parameters, you can create a custom dialog, but for all other cases, use `Base_viewWorkflowActionDialog`

Rule: Every workflow should define several worklists.

Rationale : Worklists have to be configured so that a user should just have a look to his "my favourites" menu to know what he has to do in ERP5.

The order of the worklist has to be defined according to the lifecycle of the related documents.

It is also important to use security correctly, for a validation_workflow if you make a worklist of all states in draft state that have to be validated by the Assignee, specify Assignee in the "Role(s)" guard and add `&local_roles=Assignee` in the URL.

Also, whenever it makes sense, worklists names should describe what the user have to do (good example: "Documents to validate"), and not what documents are selected (bad example: "Documents in draft state").

We usually create one worklist for each step in the workflow.

2.6 Security

In ERP5 Security model, security is splitted in two parts, one is to have 5 generic roles, and to use workflows to define security for thoses 5 generic roles (and in some cases also Owner or Anonymous roles), the second part of security definition is to define the mapping between specific business rules based on documents categories or user assignments (usually group, site and function) and those roles.

Generic business templates will have to contain security for 5A roles in workflows (states security and transitions). Project specific BT or configuration BT will contains roles for portal types based on some group, site and function categories that will have to be included in this (or another) specific BT.

For more information on security, refer to the following documents:

<http://www.erp5.org/ERP5SecurityModel>

<http://www.erp5.org/HowToDesignSecurity>

http://en.wikibooks.org/wiki/ERP5_Handbook/Magic_Security

2.6.1 Modules

Rule: Inside the Roles Tab, define the Author and Auditor roles according to the use cases

Rationale: In ERP5, all documents should follow the 5A Security model in which users are associated with roles in a per-module and/or per-document basis

An Author role should be assigned to all persons who have the right to create a new document inside a module. Those persons should also have an Auditor role because an Auditor is allowed to view documents, but not to create them.

To avoid the repetitions in the roles, we can define many roles for the same "type" of person. For that, we just need to separate the roles by a ";" for example, in the field Role, we will have Author;Auditor. All the persons who have the possibility to access to a document to validate it or to modify it after it has been created should have an Auditor role on the module.

Rule: Include the module's local roles in the business template

Rationale: When the business template is installed, roles defined on the module will not create local roles automatically, they must be explicitly included in the BT using "Local Roles" from "Detail" view.

2.6.2 Documents

Rule: Inside the Roles Tab, define the Assignor, Assignee and Associate roles and sometimes Auditor according to the use cases,

Rationale: In ERP5, all documents should follow the 5A Security model in which users are associated with roles in a per-module and/or per-document basis

Most of the times, the user who can create a document will have an "Assignor" role and the user who have to validate that document will have an "Assignee" role.

Once all the roles are correctly defined on the portal type, you should do an "Update roles settings" so that all existing documents get the correct local roles.

Rule: If the type of document should not acquire local roles from the parent documents (and modules), uncheck the "Acquire Local Roles" on this portal type configuration

Rationale: If the security of sub-document is different, you usually don't want to acquire all roles from the parent

For example, this is the case for Assignment documents, a user can be assigned some person documents, but we don't want this user to open assignment which are physically contained in this person.

3 What are different action categories and how are they used on the front page

This section is an overview of different tabs in the new front page. It is targeted at developers so that all developers agree on what are the different action categories and new actions are classified with consistency. It also aims at defining the difference between report, print and exchange actions.

3.1 Quick Search

This is a global search that supports scriptable keys and advanced search features.

This action doesn't have to be limited to document management system.

3.2 Contribute

This is to easily upload a document in the system. Just upload your file, and system will automatically guess what type of document should be created in which container, creates the document and then set some properties based on the file name and the document content. Note that using WebDAV on portal_contributions works exactly the same way.

This action could be also be used outside the scope of document management system, for instance uploading a vCard could create (or update) a person docu-

ment in person module.

3.3 Browse

This is the list of modules, grouped by business application category.

Business Application category contains entries such as “CRM”, “Accounting”, and modules are member of this category; this means that `base_application` category should be associated with the module portal type. When viewing the module, you can use the “Module Property” action to set this category membership.

3.4 New

This is to create a new document in a module. It's a way for users to create directly empty documents or document from templates.

3.5 Dig

This tab is used to search content inside a module. It shows `object_search` actions that are defined for each module.

3.6 Reports

This tab shows reports that are available for a module. This is `object_report` actions.

A report usually presents more information that what we have in the user interface, or present the information in a synthetic view. The difference between report and print actions is:

print actions are for documents that will actually be printed on paper. There are two different uses:

- the user views a page, and want to print it. It's "print what you see".
- the user have to print a document in a repeated way, like print an invoice, or print a barcode to put on packing list.

report actions are all other reports showing information. A report can be printed, so if you are unsure whether this document is a document that will always be printed or not, choose `report_action`.

Reports should never be rendered directly, an intermediate dialog is here so that the user can select which information will be used. For example, on a "total sales per region and per products" report, there should be a dialog where you choose the region(s) and product(s), a range of dates, a list of simulation states.

If the report is bound to a context (eg. latest invoices for an organisation), it's a report action for the document portal type. If the report cannot be bound to a specific context (eg. latest invoices for a region), it should be a report action on the module for this business application (in this case, on the module containing invoices).

In some cases, reports may take into account what is currently selected by the user. For example, on person module we could have a report showing the quantities of products that have been shipped to persons selected in the module list.

3.7 Printouts

This tabs shows different prints for a module.

A print action is a way to get information about the current document(s) in a format ready to be put on paper. For instance, print on a sale invoice is an invoice document ready to be printed and sent to a customer. In a module, you can for example select multiple invoices and get a document containing all invoices, one per page. Another print action could show you a list of all invoices, with the columns you have selected. Those two actions will be global print actions available on all modules.

A print action should always have a dialog, where at least you choose if you want to print in landscape or portrait mode, or even sent to printer directly (something similar to desktop applications print dialogs). If your print action is not a dialog, a default intermediate dialog will be added automatically by the page template.

View mode print actions simply show information on the current document in a way which is suited for printing. This information can include data which is not

technically properties of the document. List mode print actions always take into account what is currently selected by the user, if nothing is selected, it prints everything (which can be a problem, because it can crash the server).

In most cases, a list mode print action depends on what the user selected, so having a direct print without going through the module and checking what is selected is not useful. Exceptions can be print a catalogue of products, print a book of today's checks ... ie. something that is meant to be printed on paper but cannot be bound to one document, in that case, it should be print actions on the module. The front page print action will be visible only if some module define some custom print actions, this means that it will not be visible by default.

erp5_odt_style is the default choice for print and report actions.

3.8 Exchange

This tab shows Import/Export actions (actions from object_exchange category).

Import/Export are actions that export/import raw data to/from another application. For instance, exporting accounting transactions as spreadsheet, export one or multiple tasks as iCalendar or importing persons from vCard or Idiff are all exchange actions.


The difference between report and exchange actions is that report action produces data for human, whereas report actions produces data for machines.

From list mode, an export exchange action can use the current selection, and have a dialog for parameters (eg. the export format). Import action generally does not depend on what is selected, and also have a dialog.

erp5_ods_style or erp5_ical_style are some export styles.

Report and exports are sometimes very similar, for instance, you may want to print a report of all persons with their addresses and emails, or export this list to a spreadsheet or in an external application's format. This uses exactly the same code, only the style is different. Some exchange actions could be some report actions using ods style by default. Accountants print their trial balance (object_report) or export it to spreadsheet (object_exchange).

It's possible to add export actions from report actions, by forcing the destination skin, like:

	<div>006</div> <div>Guidelines on module creation</div>	<div>Doc. No.: 005</div> <div>Issue: 2 Rev.: 366</div> <div>Date: 01/07/08</div> <div>Page: 33 / 36</div>
---	---	---

string:\${object_url}/Module_viewMyReportDialog?your_portal_skin=ODS

3.9 Express Support

This tab only exists for ERP5 Express customers. It allows customer to access ERP5 Express services, like browse the knowledge base, access their support requests or invoices.

4 How to use proxify tool and migrate existing business templates to proxyfields

This chapter describes the procedure that has been used to convert `erp5_base` business template to use proxyfields. The goal is to have configuration bits shared by multiple fields defined in only one place.

1. Think how to group fields in field libraries

Look at existing document types and find out what information is shared by those documents.

For example, Organisations, Person both share the concept of geographical address, which is the fields city, postal code or region. Those fields are also used by Address portal type itself.

2. Create conceptual field libraries

In our address example, you should create a field library named `Address_view-FieldLibrary`, containing fields like `my_city` or `my_region`. Fields in the field library will be proxyfields to level 0 fields, from `Base_viewFieldLibrary`, which defines the default look of the fields.

For this you should use fields from existing business forms as a starting point, for example, you can copy and paste `Address_view/my_city` as `Address_view-FieldLibrary/my_city` and then use proxify tool on `Address_viewFieldLibrary` to

make my_city a proxyfield of the default string field from Base_viewFieldLibrary.

Fields in those “conceptual” field libraries must usually define title and description.

3. Use field libraries

Replace all fields in your business forms by fields from conceptual field library you just created. For this you should also use the proxify tool. Fields in those forms (like Organisation_view) should delegate almost all their values, especially title and description which must be defined in only one place. The proxify tool doesn't delegate values that are different from the original field than the target relation field.

For rare cases where the field is not shared by any other forms, you can make a proxyfield directly to the first level field from Base_viewFieldLibrary.

4. Check the result

The important rule is that your proxyfields must delegate everything possible. Everytime a value is not delegated, it's an exception, and it should have an explanation. You can use this code to display all non delegated values:

```
print '<html>'
skin_folder = context.getPortalObject().portal_skins.erp5_base # <- XXX configure this
for field_id, field in skin_folder.ZopeFind(skin_folder, search_sub=1,
                                             obj_metatypes=('ProxyField',)):
    form = field.aq_parent
    if form.getId().endswith('FieldLibrary'):
        continue

    field_path = '%s/%s' % (form.getId(), field.getId())
    print '<a href="%s/%s/manage_main">%s</a><br/>' % (
        skin_folder.absolute_url(), field_path, field_path)

    tf = field.getRecursiveTemplateField()
    for v in tf.values:
        if not field.is_delegated(v):
            print "&nbsp;&nbsp;&nbsp;%s<br/>" % v
```

print
return printed