

openCRX Basic Customizing Guide

Version 2.5.2



www.opencrx.org

WORK in PROGRESS

23-Sep-2009 @ 02:19:08 PM

License

The contents of this file are subject to a BSD license (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.opencrx.org/license.htm>

Copyright 2009 © CRIXP Corp. All rights reserved.



Table of Contents

1	About this Book.....	6
1.1	Who this book is for.....	6
1.2	What do you need to understand this book.....	6
1.3	Tips, Warnings, etc.....	6
2	Prerequisites.....	7
3	Adapting the openCRX HTML GUI to Your Needs.....	8
3.1	Overview openCRX GUI Types.....	8
3.2	Customizing Options openCRX HTML GUI.....	9
3.3	Things to do before you start Customizing openCRX.....	10
3.4	Limitations of the generic HTML GUI.....	11
3.4.1	Role-based UI.....	11
3.4.2	Model Permissions.....	11
4	Important Hints.....	12
4.1	Overloading.....	12
4.1.1	File Overloading at the Project Level.....	13
4.1.2	UI Configuration Overloading	13
4.1.3	Code Table Overloading.....	13
4.2	“Adding” Fields / Extending Objects.....	14
4.2.1	User-definable attributes of CrxObject.....	14
4.2.2	Property Data Bindings.....	15
5	Managing Packages.....	16
5.1	Enabling/Disabling Packages at the Application Level.....	16
5.2	Enabling/Disabling Packages at the User Level.....	17
6	Managing Locales.....	18
6.1	Enabling/Disabling Locales at the Application Level.....	18
6.2	Setting the Default Locale at the User Level.....	19
7	CSS, Headers, Footers, etc.....	20
7.1	Cascading Style Sheets.....	20
7.2	HTML Header and Footer Files.....	21
8	UI XML Files.....	23
8.1	Overview.....	23
8.2	UI Configuration Overloading.....	23
8.3	Inspector.....	23
8.4	Attribute Pane.....	24
8.4.1	Tabs.....	24
8.4.2	Field Groups.....	24
8.4.3	Fields / Attributes.....	25
8.5	Grid Panes.....	26
8.5.1	Expanding/Collapsing Tabs.....	27
8.5.2	AdditionalElementDefinition.....	28

8.5.3	Base Filter for Grid Tabs.....	28
8.5.4	Selection of Filterable Attributes (maxMember).....	28
8.5.5	Selection of Visible Attributes (showMaxMember).....	28
8.5.6	Advanced Attribute Selection (showMemberRange).....	28
8.6	Various XML Tags Explained.....	28
8.6.1	active.....	28
8.6.2	backColor.....	28
8.6.3	changeable.....	28
8.6.4	color.....	29
8.6.5	columnBreak.....	29
8.6.6	defaultValue.....	29
8.6.7	eventHandler.....	29
8.6.8	filterable.....	29
8.6.9	iconKey.....	30
8.6.10	inPlace.....	30
8.6.11	label.....	30
8.6.12	mandatory.....	30
8.6.13	minValue.....	31
8.6.14	maxLength.....	31
8.6.15	maxMember.....	31
8.6.16	maxValue.....	31
8.6.17	order.....	32
8.6.18	orderFieldGroup.....	32
8.6.19	orderObjectContainer.....	32
8.6.20	showMaxMember.....	32
8.6.21	showMemberRange.....	32
8.6.22	skipRow.....	32
8.6.23	spanRow.....	32
8.6.24	sortable.....	32
8.6.25	toolTip.....	32
8.7	DataBindings.....	33
8.7.1	PropertyDataBinding.....	33
8.7.1.1	A comprehensive example with PropertyDataBindings.....	33
8.7.1.2	BooleanPropertyDataBinding.....	40
8.7.1.3	IntegerPropertyDataBinding.....	40
8.7.1.4	DecimalPropertyDataBinding.....	40
8.7.1.5	StringPropertyDataBinding.....	40
8.7.1.6	DatePropertyDataBinding.....	40
8.7.1.7	DateTimePropertyDataBinding.....	40
8.7.1.8	ReferencePropertyDataBinding.....	40
8.7.2	DataBinding ProductConfigurationSet.....	41
8.7.3	DataBinding ProductConfigurationTypeSet.....	41
8.7.4	DataBindings with Referenced Objects.....	41
8.7.5	DataBindings with Composite Objects.....	41
8.7.6	DataBindings for Addresses.....	41

8.7.6.1	EmailAddressDataBinding.....	41
8.7.6.2	PhoneNumberDataBinding.....	41
8.7.6.3	WebAddressDataBinding.....	41
8.7.7	DataBindings Example Grid:.....	41
8.7.8	DataBinding AssignedActivityGroupsDataBinding.....	42
8.7.9	DataBinding DocumentFolderAssignmentsDataBinding.....	42
8.7.10	DataBinding FilteredActivitiesDataBinding.....	42
8.7.11	DataBinding FormattedNoteDataBinding.....	42
9	Code Table XML Files.....	43
9.1	Overview.....	43
9.2	Code Table Overloading.....	43
9.2.1	Adding Codes to Existing Code Tables.....	43
9.2.2	Disabling Existing Codes.....	43
9.2.3	Replacing Existing Code Tables.....	43
9.3	Segment-Specific Code Tables.....	43
10	Groovy Controls.....	44
10.1	MenuOps – openCRX Operations Menu.....	44
10.2	Navigation – openCRX Breadcrum.....	44
10.3	North – openCRX Header.....	44
10.4	RootMenu – openCRX Top Level Tabbed Menu.....	44
10.5	RootPanel – openCRX Top Level PopUp Menu.....	44
10.6	Search – openCRX Index-based Search.....	44
11	JSP Wizards.....	45
12	Forms.....	46
13	Layout JSPs.....	47
13.1	show-Default.jsp.....	47
13.2	edit-Default.jsp.....	47
13.3	Custom Layout JSPs.....	47
14	Advanced Customizing Options.....	48
14.1	Workflows.....	48
14.2	Java Controls.....	48
14.3	Portal Extensions.....	48
14.4	UML Model Extensions.....	48
14.5	Application Logic Extensions.....	48
15	Other Customizing Options.....	49
15.1	web.xml.....	49
15.1.1	uiRefreshRate.....	49
15.1.2	load-on-startup.....	49
15.1.3	session-timeout.....	50
15.1.4	transport-guarantee.....	50
16	Next Steps.....	51

List of Figures

Figure 1: Types of openCRX GUIs.....	8
Figure 2: Non-Java GUIs for openCRX.....	8
Figure 3: openCRX Servlets enabling access with third-party clients.....	9
Figure 4: Customizing Options – openCRX Standard HTML GUI.....	9
Figure 5: openCRX Development and Customizing with Custom Project.....	12
Figure 6: UML Model - CrxObject.....	14
Figure 7: Add the string field Y!M nick to Contact objects.....	14
Figure 8: Launch Wizard User Settings.....	17
Figure 9: Wizard User Settings – enable/disable Root Menu Entries.....	17
Figure 10: Set Default Locale by Saving User Settings	19
Figure 11: Placement of attributes in the attribute pane (Tab, Group, Position)	25
Figure 12: Placement of grid tabs (Pane, -, Position).....	26
Figure 13: Placement of attributes in grids (Level 0, Level 1, Level 2).....	27
Figure 14: Sample Extension of class Product with PropertyDataBindings.....	37
Figure 15: Sample Extension of class Product with PropertyDataBindings.....	38
Figure 16: Property Set Extension.....	39

List of Listings

Listing 1: UI Customizing File zorder_contact.xml.....	15
Listing 2: List of Packages in web.xml.....	16
Listing 3: Enabling/Disabling Packages in web.xml.....	16
Listing 4: Locales in web.xml.....	18
Listing 5: Activating/Deactivating Locales in web.xml.....	18
Listing 6: Active Locales in Login.jsp.....	19
Listing 7: UI Customizing File zorder_product.xml.....	33
Listing 8: Example AssignedActivityGroupsDataBinding.....	42
Listing 9: uiRefreshRate in web.xml.....	49
Listing 10: load-on-startup in web.xml.....	49
Listing 11: sesstion-timeout in web.xml.....	50
Listing 12: transport-guarantee in web.xml.....	50

1 About this Book

This book describes various ways of customizing the openCRX Standard HTML GUI to adapt the look and feel to your personal tastes and preferences or to your company's CI (corporate identity).

openCRX is the leading enterprise-class open source CRM suite. openCRX is based on openMDX, an open source MDA framework based on the OMG's model driven architecture (MDA) standards. This guarantees total openness, standards compliance, a state-of-the-art component-based architecture, and virtually unlimited scalability.

1.1 Who this book is for

The intended audience are openCRX administrators and advanced users.

1.2 What do you need to understand this book

It is helpful to have a good understanding of the openCRX architecture. We assume that you are able to read/understand the openCRX UML models and the openCRX Javadoc (Java API). It is also assumed that you are familiar with XML files and you should know how to program JSPs.

1.3 Tips, Warnings, etc.

We make use of the following pictograms:



Tip

Information provided as a "Tip" might be helpful for various reasons: time savings, risk reduction, etc. - it goes without saying that we advise to follow our guides meticulously

meticulous \muh-TIK-yuh-luhs\, *adjective*:
Extremely or excessively careful about details.



Important

You should carefully read information marked with "Important". Ignoring such information is typically not a good idea.



Warning

Warnings should not be ignored (risk of data loss, etc.)

2 Prerequisites

To work through some of the examples and in particular to build your own customized openCRX, there are some prerequisites:

- openCRX Server (v2.5.2 or newer).
You can either follow the openCRX Server Installer documentation at <http://www.opencrx.org/server.htm> or you can do a manual installation of openCRX following the guide openCRX Manual Installation Tomcat 6.
- openCRX SDK (v2.5.2 or newer).
Follow the openCRX SDK Installer documentation at <http://www.opencrx.org/sdk.htm>.
- Most of the openCRX customizing files are UTF-8 encoded (see Wikipedia for information on UTF-8 encoding) and if you intend to edit such files you **must use an UTF-8 enabled editor**. Many simple text editors work fine (e.g. Notepad on Windows, or gedit on Linux), but make sure you test your editor of choice before you waste a lot of time and end up with ruined files that are not properly encoded anymore.
- While you can manage all your changes “manually”, we recommend the use of an openCRX custom project. If you manage your changes and extensions to the standard distribution of openCRX with a custom project, you will be able to easily migrate your changes/extensions to new versions of openCRX. Furthermore, you can easily build customized openCRX EARs. Instructions on how to create an openCRX custom project are available from the openCRX Wiki (see “How to create custom projects”).

Throughout this guide we assume that the custom project is called **sample** and the data directory is **org.opencrx.sample**. Hence, most of the files you will be changing/creating in this guide are contained in one of the subdirectories of the directory

```
...\opencrx-custom\sample\src\data\org.opencrx.sample
```

If you decide to manipulate files directly in the apps folder, the same files are contained in one of the subdirectories of the directory

```
...\apache-tomcat-6\apps\opencrx-core-CRX
```

3 Adapting the openCRX HTML GUI to Your Needs

3.1 Overview openCRX GUI Types

openCRX is distributed with a generic HTML GUI (based on openMDX/portal) that connects to the openCRX API as shown below:

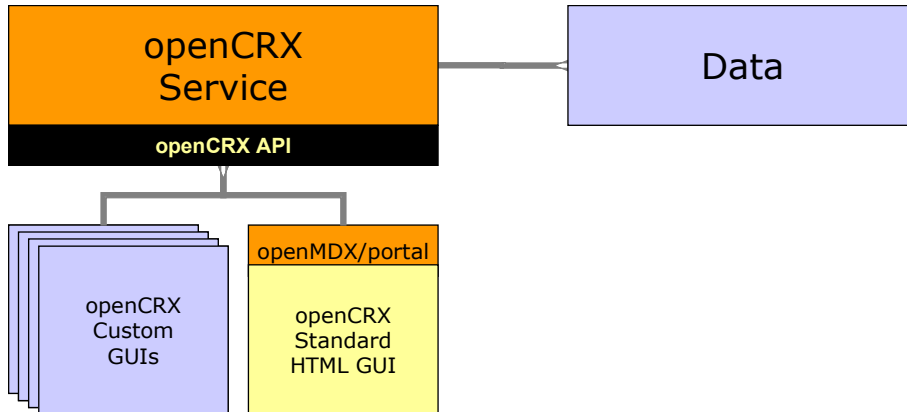


Figure 1: Types of openCRX GUIs

This Ajax-enabled HTML GUI supports a wide range of modern browsers, including Firefox, Opera, Safari, Chrome, KDE Konqueror, and IE.

It goes without saying that you can also program your own custom GUI. openCRX/store is an example DHTML GUI consisting of a set of manually programmed JSPs connecting to the openCRX API.

It is also possible to develop non-Java-based GUIs for openCRX. You could – for example – write an XML-RPC Adapter (see Apache XML-RPC):

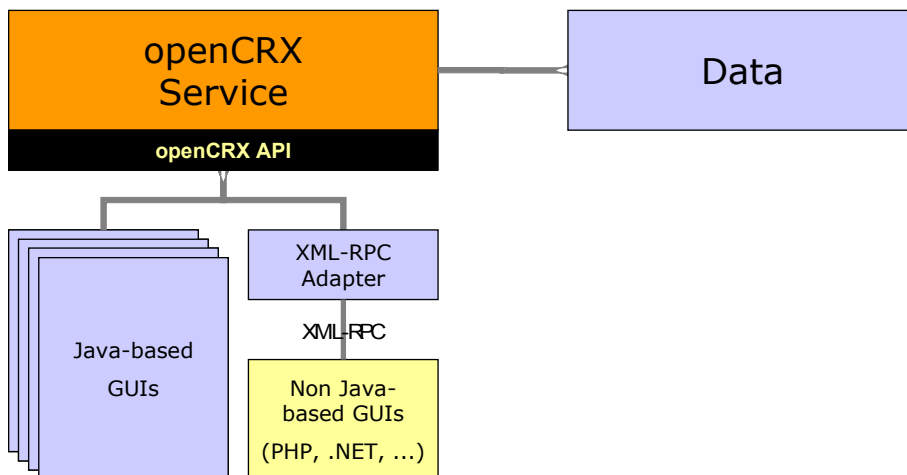


Figure 2: Non-Java GUIs for openCRX

Another option you might consider is REST (Representational State Transfer); see also <http://www.opencrx.org/opencrx/2.3/new.htm#REST> and the openCRX Wiki (“How to use the REST servlet”).

openCRX includes a set of servlets (caldav, ical, imap, vcard, rest, news, ...) allowing you to connect to openCRX with a wide range of specialized third-party clients like Mozilla Thunderbird, MS Outlook, iPhone, and others:

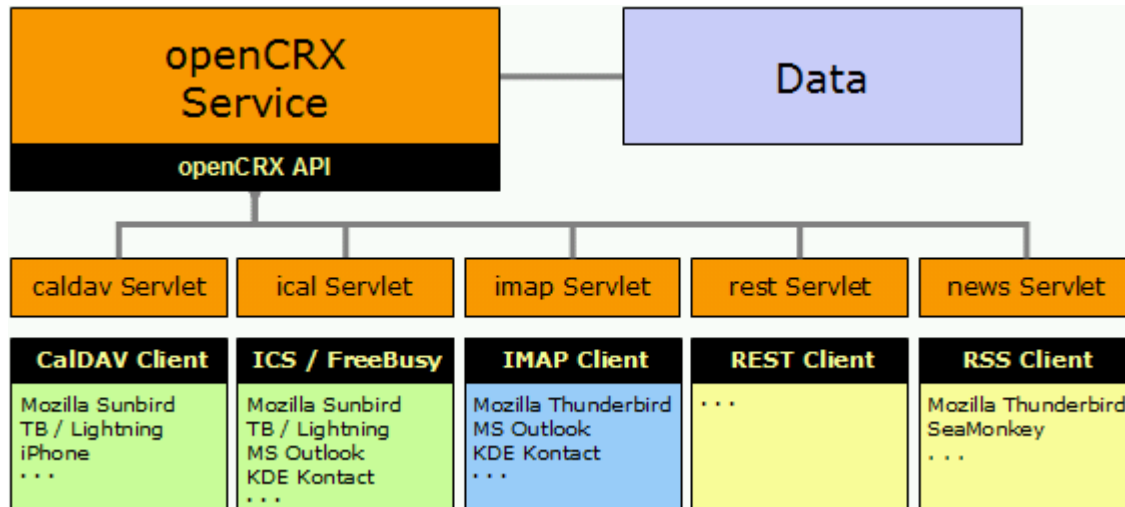


Figure 3: openCRX Servlets enabling access with third-party clients

3.2 Customizing Options openCRX HTML GUI

The openCRX Standard HTML GUI can be customized in many ways to suit your needs:

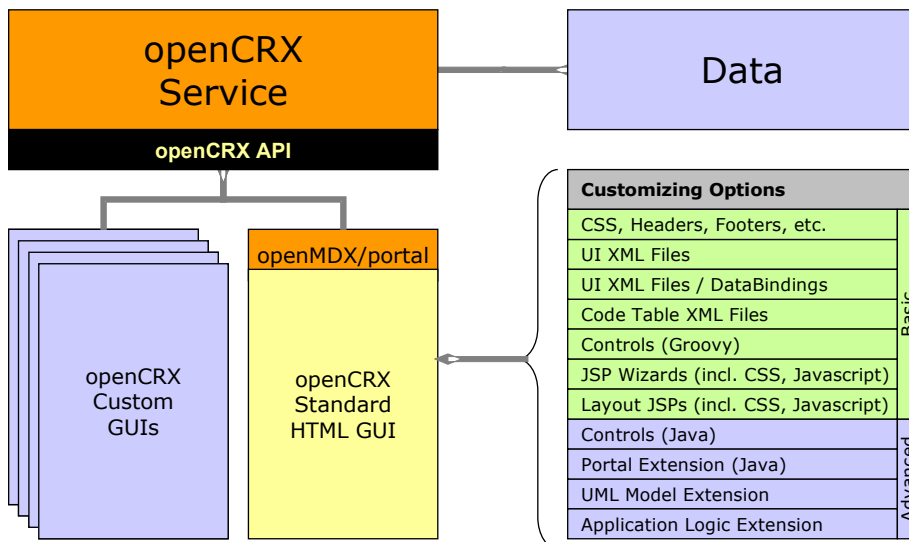


Figure 4: Customizing Options – openCRX Standard HTML GUI

Basic customizing options are relatively straight-forward and require no (or only moderate) programming know how. Advanced customizing options, however, require a good understanding of the openCRX architecture. This guide covers basic customizing options. If you're interested in advanced customizing options, have a look at the source code or consider attending an openCRX Developer Workshop (see www.opencrx.org for more information).

3.3 Things to do before you start Customizing openCRX

Adapting openCRX to your needs typically involves the following steps:

1. **Collect the requirements** of the (customized) application – this step requires a solid understanding of the business domain:
 - your company's CI information (e.g. color schemes, fonts, etc.)
 - business objects and their attributes
 - mock screens or screen shots of existing applications
 - typical use cases, workflows, etc.
2. **Map the business requirements to openCRX** – this step requires know how across both your business domain and the openCRX domain:



Note that openCRX objects are typically normalized, i.e. it is not uncommon that a single business object gets spread out across multiple openCRX objects. In other words, a single business object and its attributes often times do not map to a single openCRX object and its attributes.

Example: mapping the data of a contact as it is available on a business card typically involves multiple openCRX objects like

- ◆ Contact (first/name, ...) or LegalEntity (company name)
- ◆ PostalAddress (street, city, zip code, ...)
- ◆ PhoneNumber (country code, number, extension, ...)

Hence, study the openCRX UML models before you start the mapping process. Take your time as the openCRX UML models are very comprehensive; focus on those aspects of the model that are most relevant to your project, e.g. account1, activity1.

- map business objects to (possibly a list of) openCRX objects; depending on your business domain you might have to be a little inventive and maybe adapt/change the meaning of some of the openCRX objects
- map each attribute of your business objects to an attribute of an openCRX object



If you come to the conclusion that you need additional attributes in certain openCRX objects, do not jump at extending the openCRX UML model (such an undertaking requires a solid understanding of the openCRX architecture and the build processes). Instead, learn about and understand the following mechanisms for “adding” additional fields to an object:

- ◆ User-definable attributes of CrxObject
- ◆ DataBinding Property

3. **Customizing the openCRX Standard HTML GUI** – that is what this guide is all about...

Obviously, if you're only interested in changing some of the colors, there is not much to do in step 2 above. However, if you plan to capture your companies business objects, business processes, forms, etc. with openCRX, it pays off to spend some time on collecting the requirements and then properly map them to openCRX before you get down to customizing...



It is a good idea to create an openCRX custom project before you get started with customizing openCRX. Like this you will be able to collect all your changes and enhancements in one place and it will be easy to create custom EARs which you can readily deploy.

Instructions for creating an openCRX custom project are available from the openCRX Wiki.

3.4 Limitations of the generic HTML GUI

Even though the openCRX HTML GUI is extremely flexible, we would like to point out a few limitations (not due to bad design, but rather we are looking at advanced features that have not been implemented yet).

3.4.1 Role-based UI

openCRX features UI perspectives, a mechanism that enables users to have different UI customizations based on the current role (e.g. one GUI for sales and another GUI for the back office). More information is available from <http://www.opencrx.org/opencrx/2.3/new.htm#UIPerspectives>.

Obviously, the same goal can also be achieved with multiple web applications. Simply create a custom project for each role / customization and then deploy multiple web applications.

3.4.2 Model Permissions

Model permissions are not implemented yet. Hence you cannot control access to individual attributes of an object with simple customization (the openCRX security plugin controls access to complete objects, not to individual attributes of an object). You can, however, deploy multiple web applications or work with Layout JSPs to achieve the same goal.

4 Important Hints

The following information is so important that it deserves its own chapter!

4.1 Overloading

Overloading is a concept that allows you to selectively enhance (or even replace) features of the standard distribution of openCRX with your own changes and/or extensions.



The architecture of openCRX is such that you should be able to add your own customizing and extensions without actually changing any of the core files of the distribution. The big advantage of leaving core files unchanged and keeping all your changes/extensions separate is **release capability**, i.e. instead of creating and then maintaining your own openCRX branch (which you should really try to avoid) you keep your changes and extensions in a custom project.

The reason is the following one: **maintaining your own openCRX branch requires advanced know how**, i.e. it is difficult and very time-consuming, whereas **maintaining your own custom project is easy** because upgrading your custom project to a new openCRX version typically requires minor changes only (if any at all).

Do not go for quick fixes by changing files of the core distribution as you are guaranteed to run into problems down the road. Make use of **openCRX custom projects** and use the power and the flexibility of the openCRX SDK and the various **overloading concepts**.

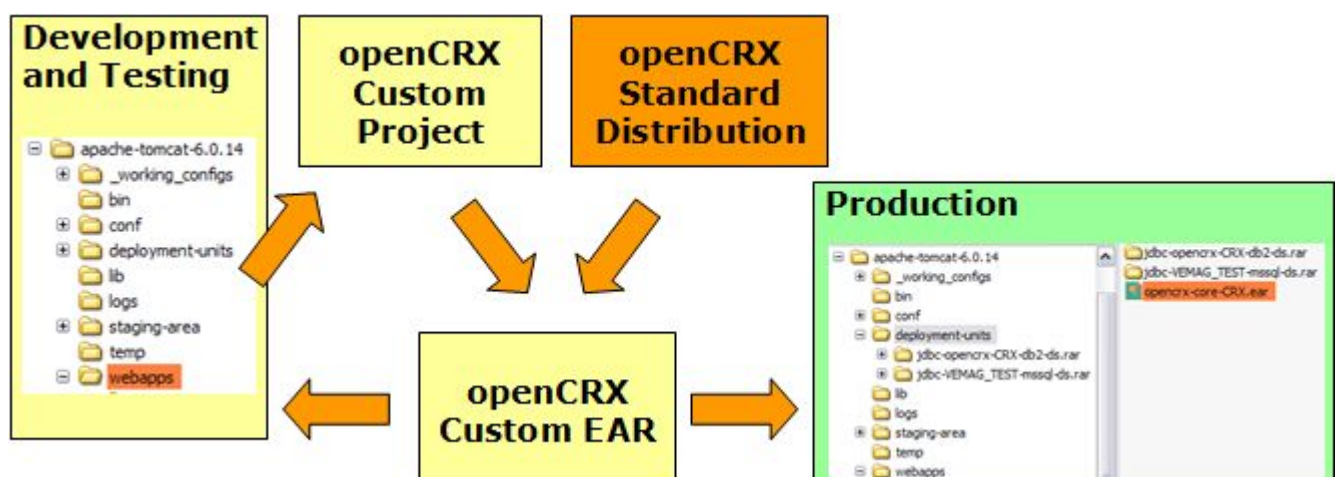


Figure 5: openCRX Development and Customizing with Custom Project

4.1.1 File Overloading at the Project Level

Typically, files in your custom project will be **added** to the custom EARs. However, if a file in your custom project has the same name as a file of the standard distribution, your file will actually **replace** the respective file of the standard distribution when you build custom EARs.

Example: Let's suppose you want to add some fancy CSS to openCRX. If you call your file `myFancy.css` and put it into the directory `sample\src\data\org.opencrx.sample_style` your css file will be **added** to the various css files that already exist when you build your custom EARs. If you call your file `colors.css`, however, your file will **replace** the file with the same name that is contained in the standard distribution of openCRX (and unless you know what your doing it is quite likely that the coloring of openCRX will look strange with your custom EARs...).

4.1.2 UI Configuration Overloading

It is likely that you want to change some of the default customizing. Quite possibly, however, (a) you want to make a few changes only and (b) you want to keep these changes if you upgrade to a new version of openCRX. This is where UI configuration overloading can add value. Instead of changing the original UI configuration files provided with the standard distribution you create a new configuration file (or multiple configuration files) containing all your changes. Make sure that you name your file(s) containing changed UI Element Definitions such that your changes are loaded **AFTER** the default configuration files, thereby overloading the original configuration.

More information is available in chapter 8.2 UI Configuration Overloading.

4.1.3 Code Table Overloading

Similar to UI configuration overloading, you can also overload code tables. Instead of changing original code tables you create new code table files that contain your changes/extensions. It is also possible to deactivate codes of the standard distribution. Make sure that you name your file(s) containing changed code tables such that your changes are loaded **AFTER** the default code tables, thereby overloading the code tables.

More information is available in chapter 9.2 Code Table Overloading.



UI configuration files and code table files are loaded in **alphabetical order**. Hence, if the same element is defined multiple times, the definition contained in the file that is (in alphabetical order) loaded last wins.

4.2 “Adding” Fields / Extending Objects

Even though the openCRX UML Model covers a wide range of use cases, you may still come to the conclusion that you have to “add” a field to a particular object. While the newbie's approach would be to add a field to the appropriate database table and then patch the code a bit here and there, take our advice and make use of the advanced extension mechanisms provided by openCRX.

4.2.1 User-definable attributes of CrxObject

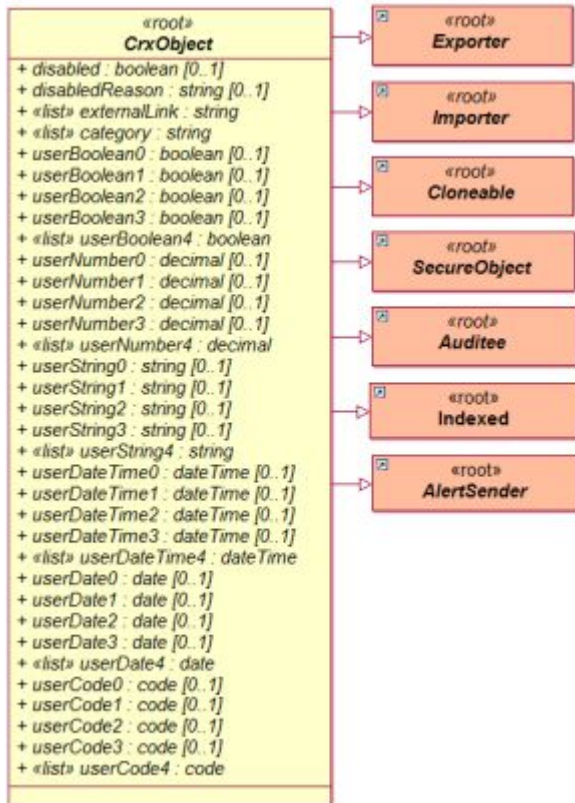


Figure 6: UML Model - CrxObject

The vast majority of openCRX's objects are CrxObjects, and hence they feature user-definable attributes as shown in the figure on the left. There are four single-valued attributes and one multi-valued attribute of each type:

- boolean
- decimal
- string
- dateTime
- date
- code

In the standard distribution of openCRX these attributes are disabled (i.e. set to `<active>false</active>`). You can verify this by looking at the UI customizing file `.../config/ui/Root/en_US.xml` (search for `CrxObject:user`).

Note that all of these user-definable attributes are already available in the DB schemas, i.e. they are ready to be used, you just have to enable them.

To make an example, let's assume your Contacts absolutely need another String field to store the Yahoo! Messenger Nickname as shown below:

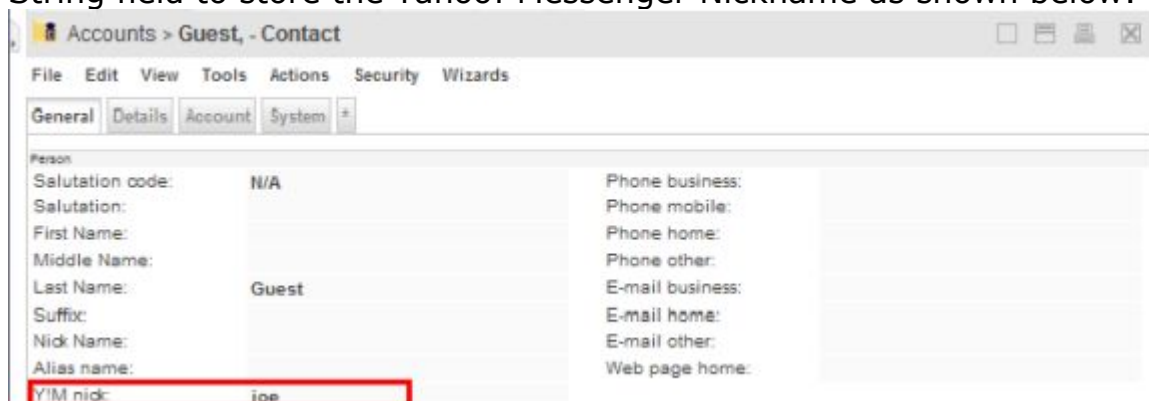


Figure 7: Add the string field Y!M nick to Contact objects

All you have to do to “add” the field Y!M nick to Contacts is to deploy the following UI customizing file **zorder_contact.xml** to the directory `{TOMCAT_INSTALL_DIR}/apps/opencrx-core-CRX/opencrx-core-CRX/WEB-INF/config/ui/Root/en_US` and then restart Tomcat.

Listing 1: UI Customizing File `zorder_contact.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<org.openmdx.base.Authority xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="org.openmdx:ui1"
xsi:noNamespaceSchemaLocation="xri:+resource/org/openmdx/ui1/xml/ui1.xsd">
  <_object/>
  <_content>
    <provider>
      <org.openmdx.base.Provider qualifiedName="CRX" _operation="null">
        <_object/>
        <_content>
          <segment>
            <org.openmdx.ui1.Segment qualifiedName="Root" _operation="null">
              <_object/>
              <_content>
                <elementDefinition>
                  <org.openmdx.ui1.ElementDefinition name="org.opencrx:kernel:account1:Contact:userString0">
                    <_object>
                      <active>true</active>
                      <toolTip>
                        <_item>Y!M nick</_item>
                      </toolTip>
                      <label>
                        <_item>Y!M nick</_item>
                      </label>
                      <order>
                        <_item>0</_item> <!-- tab -->
                        <_item>0</_item> <!-- field group -->
                        <_item>75</_item> <!-- position -->
                      </order>
                    </_object>
                  </org.openmdx.ui1.ElementDefinition>
                </elementDefinition>
              </_content>
            </org.openmdx.ui1.Segment>
          </segment>
        </_content>
      </org.openmdx.base.Provider>
    </provider>
  </_content>
</org.openmdx.base.Authority>
```

The above file is – strictly speaking – not adding a field to the class Contact, it is rather activating the attribute `userString0` of the class Contact (which it inherits from the class `CrxObject`). The field `userString0` is already available in the standard distribution of openCRX, it's just not visible by default.



Mapping your special requirements to user-definable attributes is an extremely efficient way of “extending” the standard distribution of openCRX, also in terms of performance.

4.2.2 Property Data Bindings

What can you do if you need even more user-definable attributes, i.e. more than the ones provided by `CrxObject`? This is where the `PropertyDataBinding` concept comes in handy. With `PropertyDataBindings` you get access to a virtually unlimited pool of user-definable attributes. See chapter 8.7.1 `PropertyDataBinding` for a detailed introduction into this topic).

5 Managing Packages

5.1 Enabling/Disabling Packages at the Application Level

With the openCRX standard distribution all available packages are enabled. The openCRX administrator may wish to disable certain packages at the application level if they are not used. This chapter shows how you can achieve this.

In the custom project sample, the package list is contained in the file:

`opencrx—custom\sample\src\data\org.opencrx.sample\WEB-INF\web.xml`

Once deployed on Tomcat, the package list is contained in the file

`apps\opencrx-core-CRX\opencrx-core-CRX\WEB-INF\web.xml`

Look for the section `<!-- Admin -->` to find a list of available packages:

Listing 2: List of Packages in web.xml

```
<!-- Admin -->
<init-param>
  <param-name>rootObject[0]</param-name>
  <param-value>xri:@openmdx:org.opencrx.kernel.admin1/provider/CRX/segment/${SEGMENT}</param-value>
</init-param>
<!-- Home -->
<init-param>
  <param-name>rootObject[1]</param-name>
  <param-value>xri:@openmdx:org.opencrx.kernel.home1/provider/CRX/segment/${SEGMENT}/userHome/${USER}</param-value>
</init-param>
...
```

You can disable packages by commenting them out (`<!--` to open a comment and `-->` to close a comment). The following example shows how to deactivate the package depot1:

Listing 3: Enabling/Disabling Packages in web.xml

```
...
</init-param>
<!-- Depots -->
<!--
<init-param>
  <param-name>rootObject[6]</param-name>
  <param-value>xri:@openmdx:org.opencrx.kernel.depot1/provider/CRX/segment/${SEGMENT}</param-value>
</init-param>
-->
<!-- Documents -->
<init-param>
  <param-name>rootObject[6]</param-name>
  <param-value>xri:@openmdx:org.opencrx.kernel.document1/provider/CRX/segment/${SEGMENT}</param-value>
</init-param>
<!-- Buildings -->
...
```



Warning

Please note that you must **renumber all the packages listed after the package you deactivated** so that the package numbering does not have any gaps (i.e. **numbering of active packets starts at 0 and it must be consecutive**).



Tip

It is also possible to change the order of the active packages by renumbering them. However, you must still ensure both that the numbering starts at 0 and that the numbering is consecutive.

5.2 Enabling/Disabling Packages at the User Level

Individual user can enable/disable root menu entries with the wizard **User Settings** (available on a user's Homepage):

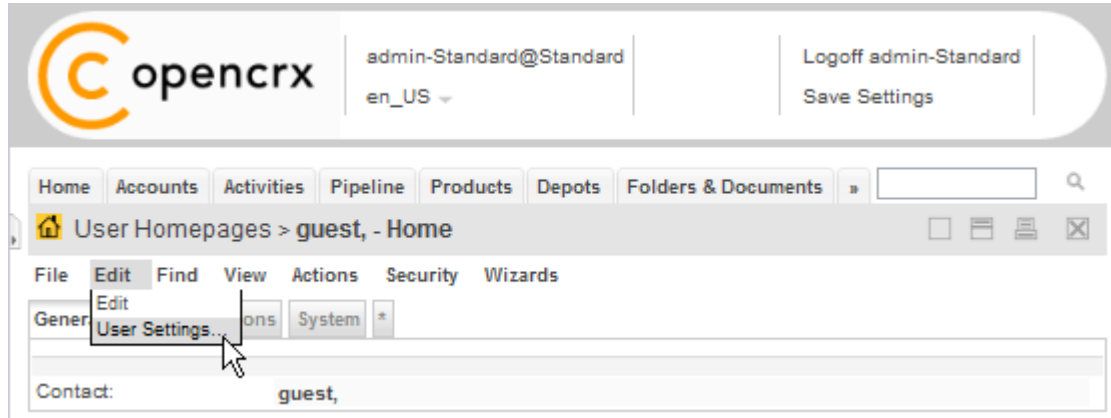


Figure 8: Launch Wizard User Settings

Once the wizard has loaded, uncheck entries you don't need:

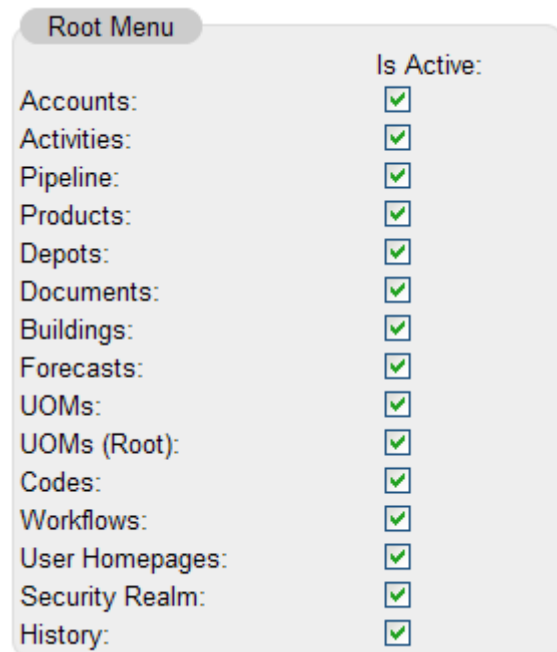


Figure 9: Wizard User Settings – enable/disable Root Menu Entries



Important

Please note that entries corresponding to packages disabled by the openCRX administrator cannot be enabled with this wizard. Packages disabled in web.xml are not available at all!



Tip

Depending on the width of your screen you can adjust the number of items shown as tabs in the top-level navigation in the same wizard (fewer items for narrow screens, more items for wider screens).

6 Managing Locales

The default installation of openCRX activates all locales that are included in the Open Source distribution. The openCRX administrator may wish to deactivate certain locales from the locale list. This chapter shows how you can achieve this.

6.1 Enabling/Disabling Locales at the Application Level

The locale list is contained in the file

`opencrx-core-CRX.ear\opencrx-core-CRX.war\WEB-INF\web.xml`

Look for the section `<!-- locales -->` to find a list of available locales:

Listing 4: Locales in web.xml

```
<!-- locales -->
<init-param>
  <param-name>locale[0]</param-name>
  <param-value>en_US</param-value>
</init-param>
<init-param>
  <param-name>locale[1]</param-name>
  <param-value>de_CH</param-value>
</init-param>
<init-param>
  <param-name>locale[2]</param-name>
  <param-value>es_MX</param-value>
</init-param>
...
```

You can deactivate locales by simply commenting them out. The following example shows how to deactivate the locale `de_CH`.

Listing 5: Activating/Deactivating Locales in web.xml

```
<!-- locales -->
<init-param>
  <param-name>locale[0]</param-name>
  <param-value>en_US</param-value>
</init-param>
<!--
<init-param>
  <param-name>locale[1]</param-name>
  <param-value>de_CH</param-value>
</init-param>
-->
<init-param>
  ...
```



Please note that you must **not** deactivate the base locale (that is the locale with the id 0, typically `en_US`) as the base locale contains a lot of customizing information not present in other locales.

As the Login page is displayed before the authentication of the user has taken place, `Login.jsp` cannot access the above information. That is why the list of

available and active locales are maintained in the file **localeSettings.jsp** as well. Changing the list of active locales is straightforward. Comment out any of the lines in the following code segment of `localeSettings.jsp` to disable the respective locale:

Listing 6: Active Locales in Login.jsp

```
...
List activeLocales = new LinkedList();
activeLocales.add("en_US");
activeLocales.add("cs_CZ");
activeLocales.add("de_CH");
activeLocales.add("es_CO");
activeLocales.add("es_MX");
// activeLocales.add("fa_IR"); this locale is not active
activeLocales.add("fr_FR");
activeLocales.add("it_IT");
activeLocales.add("ja_JP");
// activeLocales.add("nl_NL"); this locale is not active
activeLocales.add("pl_PL");
activeLocales.add("pt_BR");
activeLocales.add("ro_RO");
activeLocales.add("ru_RU");
activeLocales.add("sk_SK");
activeLocales.add("sv_SE");
activeLocales.add("tr_TR");
activeLocales.add("zh_CN");
...
```

The above example shows how to disable the locales `fa_IR` and `nl_NL` in the Login page of openCRX.



Note that you **must use an UTF-8 enabled editor to change the file `localeSettings.jsp`**. If you break the UTF-8 encoding of this file your locale drop will contain strange symbols...

6.2 Setting the Default Locale at the User Level

A user's default locale can be set by choosing/activating the desired locale and then clicking on [**Save Settings**] in the header of the application as shown below:

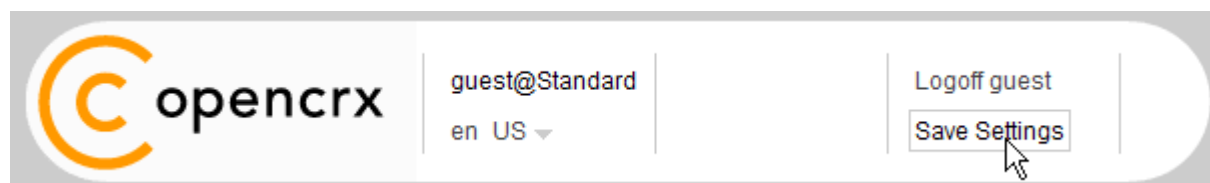


Figure 10: Set Default Locale by Saving User Settings

If the login page supports a user's preferred locale `xx_YY`, you can request the login page in that locale `xx_YY` by appending the string **"?locale=xx_YY"** to the default login URL.

Example: the following URL loads the German login page:

`http://demo.opencrx.org/opencrx-core-CRX/Login.jsp?locale=de_CH`

7 CSS, Headers, Footers, etc.

7.1 Cascading Style Sheets

The directory `...\opencrx-core-CRX_style` contains various CSS files:

CSS File	Purpose / Comments
calendar-small.css	customizing of the DHTML calendar [http://www.dynarch.com/projects/calendar/]
colors.css	various color settings (fonts, backgrounds, etc.)
default.css	none – legacy file
dummy.css	none – repository placeholder
header.css	customizing of the header
infovis.css	customizing of the JavaScript Information Visualization Toolkit (JIT) [http://blog.thejit.org/javascript-information-visualization-toolkit-jit/]
n2default.css	customizing of openMDX/portal (the default openCRX GUI)
ssf.css	customizing of the Suckerfish menus (http://www.htmldog.com/articles/suckerfish/dropdowns/)

Please note that various third-party modules (e.g. WYMeditor, wiky) may also load CSS files. Those files are typically located within the respective modules directory structure (e.g. `.../javascript/wymeditor`, `.../javascript/wiky`).



Hints on Cascading Order (source: Introduction to CSS / w3schools)

Multiple Styles Will Cascade Into One

Style sheets allow style information to be specified in many ways. Styles can be specified inside a single HTML element, inside the `<head>` element of an HTML page, or in an external CSS file. Even multiple external style sheets can be referenced inside a single HTML document.

Cascading Order

What style will be used when there is more than one style specified for an HTML element? Generally speaking we can say that all the styles will "cascade" into a new "virtual" style sheet by the following rules, where number four has the highest priority:

1. Browser default
2. External style sheet
3. Internal style sheet (inside the `<head>` tag)
4. Inline style (inside an HTML element)

So, an inline style (inside an HTML element) has the highest priority, which means that it will override a style declared inside the <head> tag, in an external style sheet, or in a browser (a default value).

Depending on the width of your screen you can adjust the number of items shown as tabs in the top-level navigation in the same wizard (fewer items for narrow screens, more items for wider screens).

As most of the styles used by openCRX are defined in external style sheets, your options to override defaults are intact.



Adding (internal and/or external) style sheets to all pages is easy as all pages are generated by the following 2 Layout JSPs:

...\opencrx-core-CRX\WEB-INF\config\layout\en_US\edit-Default.jsp

...\opencrx-core-CRX\WEB-INF\config\layout\en_US\show-Default.jsp

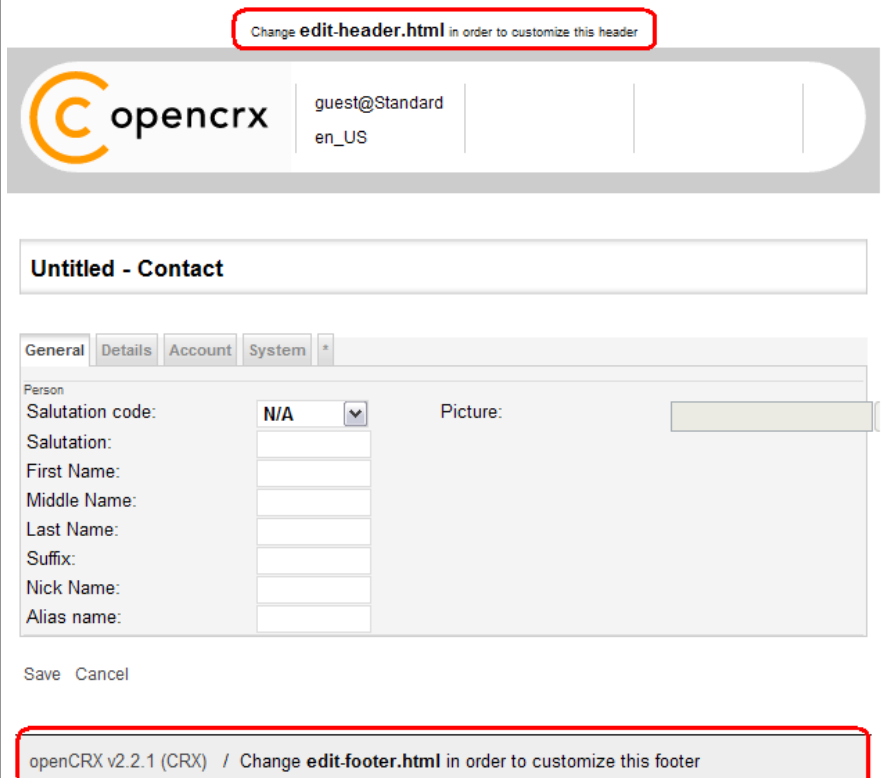
See also chapter 13 Layout JSPs for additional information.

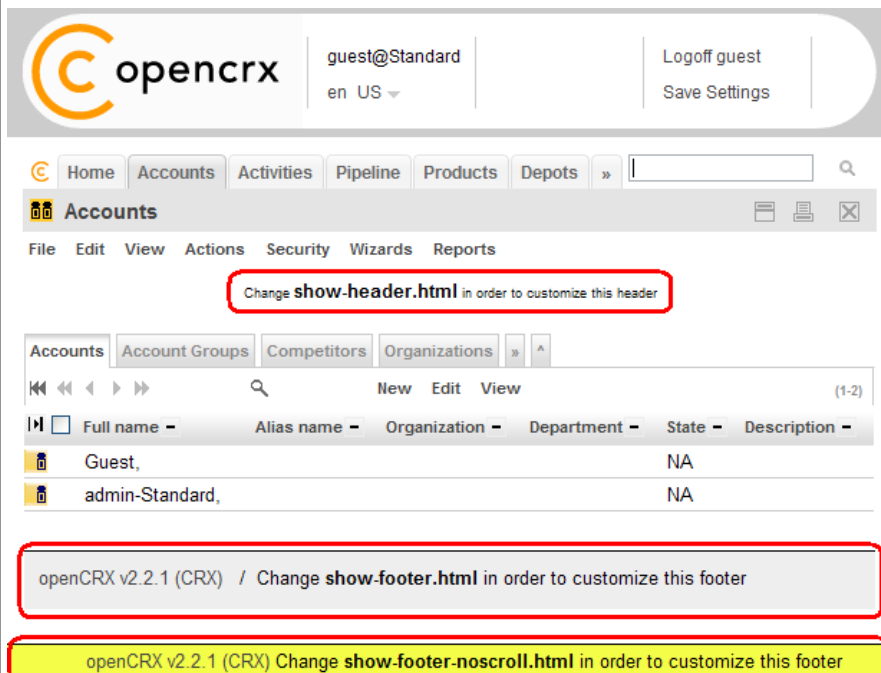
7.2 HTML Header and Footer Files

The Layout JSPs and the login page (Login.jsp) by default include various HTML files. They are located in the directory ...\opencrx-core-CRX and you can adapt them to your liking:

HTML File	Screen Shot /point of insertion
login-header.html login-note.html login-footer.html	

See also chapter 13 Layout JSPs for additional information.

HTML File	Screen Shot /point of insertion
<p>edit-header.html edit-footer.html</p>	 <p>The screenshot shows the openCRX user interface. At the top, there is a header area with the openCRX logo and user information (guest@Standard, en_US). A red box highlights the text "Change edit-header.html in order to customize this header" in the header area. Below the header is a "Untitled - Contact" form with tabs for "General", "Details", "Account", and "System". The "General" tab is active, showing fields for "Salutation code", "Picture", "Salutation", "First Name", "Middle Name", "Last Name", "Suffix", "Nick Name", and "Alias name". At the bottom of the page, a footer area contains the text "openCRX v2.2.1 (CRX) / Change edit-footer.html in order to customize this footer", which is also highlighted with a red box.</p>

HTML File	Screen Shot /point of insertion
<p>show-header.html show-footer.html show-footer-noscroll.html</p>	 <p>The screenshot shows the openCRX user interface. At the top, there is a header area with the openCRX logo and user information (guest@Standard, en_US). A red box highlights the text "Change show-header.html in order to customize this header" in the header area. Below the header is a navigation menu with items like "Home", "Accounts", "Activities", "Pipeline", "Products", and "Depots". Below the navigation menu is a "Accounts" section with a menu bar (File, Edit, View, Actions, Security, Wizards, Reports) and a table. The table has columns for "Full name", "Alias name", "Organization", "Department", "State", and "Description". The table contains two rows: "Guest, NA" and "admin-Standard, NA". At the bottom of the page, there are two footer areas. The first footer area contains the text "openCRX v2.2.1 (CRX) / Change show-footer.html in order to customize this footer", which is highlighted with a red box. The second footer area contains the text "openCRX v2.2.1 (CRX) Change show-footer-noscroll.html in order to customize this footer", which is highlighted with a red box.</p>

8 UI XML Files

8.1 Overview

8.2 UI Configuration Overloading

8.3 Inspector

The inspector shows an object with its attributes:

The screenshot displays the openCRX web application interface. At the top, there is a header with the openCRX logo, user information (guest@Standard, en_US), and navigation links (Logoff guest, Save Settings). Below the header is a navigation bar with tabs for 'My Homepage', 'Accounts', 'Activities', and 'Pipeline'. The main content area shows the breadcrumb 'Accounts > Guest, Joe O - Contact' and a menu with options like 'File', 'Edit', 'View', 'Tools', 'Actions', 'Security', and 'Wizards'. The 'General' tab is selected, showing a form for contact details. The form is organized into sections: 'Person', 'Postal Address Home / Business', 'Contact Information', and 'Job Information'. Each section contains various fields for personal and professional data, including salutations, names, phone numbers, email addresses, and addresses.

Person			
Salutation code:	Mr.	Phone business:	+41 (44) 508-9999
Salutation:		Phone mobile:	+41 (79) 444-3333
First Name:	Joe	Phone home:	
Middle Name:	O	Phone other:	
Last Name:	Guest	E-mail business:	joe.guest@opencrx.org
Suffix:		E-mail home:	
Nick Name:	jg	E-mail other:	
Alias name:	guest	Web page home:	http://www.opencrx.org/
Postal Address Home / Business			
Address line(s):	Mr. Joe Guest c/o CRIXP Corp.	Address line(s):	
Street / PO Box:	Technoparkstr. 1	Street / PO Box:	
State:		State:	
ZIP/Postal code:	8005	ZIP/Postal code:	
City:	Zürich	City:	
Country/Region:	Switzerland [CH]	Country/Region:	
Contact Information			
Preferred method:	E-Mail	No postal mail:	<input checked="" type="checkbox"/>
Spoken language:	German [de/ger/deu]	No bulk postal mail:	<input checked="" type="checkbox"/>
Written language:	German [de/ger/deu]	Do not phone:	<input checked="" type="checkbox"/>
No E-Mail:	<input type="checkbox"/>	Do not fax:	<input checked="" type="checkbox"/>
Job Title:	Director	Organization:	CRIXP Corp.
Job Role:	Chief Tester	Department:	Quality Assurance

8.4 Attribute Pane

8.4.1 Tabs

The ui file **common.xml** contains tab definitions that apply to all objects :

[General]	Tab:0	this tab typically contains the most important attributes of an object and is almost always visible
[Details]	Tab:10	this tab typically contains attributes that are used less frequently; if no attributes are assigned to this tab then it is not visible
[User]	Tab:90	by default all user defined attribute are assigned to this tab; this tab, however, is not visible by default
[System]	Tab:900100	this tab typically contains various system attributes related to object creation and object changes, security, etc.

You can define your own tabs. The following example shows the definition of the tab **[Accounts]** of the attribute pane of accounts (ui file account.xml):

Example	<pre><org.openmdx.ui1.ElementDefinition name="org:opencrx:kernel:account1:Account:Pane:Attr:Tab:25"> <_object> <active>true</active> <toolTip> <_item>Account</_item> </toolTip> <label> <_item>Account</_item> </label> </_object> <_content/> </org.openmdx.ui1.ElementDefinition></pre>
----------------	--

8.4.2 Field Groups

In the ui file common.xml there are various default field groups defined that apply to all objects, e.g. Field Groups 0, 10, 20, and 30 in the the tabs [General] and [Details].

You can define your own field groups like shown in the following example:

Example	<pre><org.openmdx.ui1.ElementDefinition name="org:opencrx:kernel:account1:Account:Pane:Attr:Tab:25:Group:10"> <_object> <active>true</active> <toolTip> <_item></_item> </toolTip> <label> <_item>&nbsp;</_item> </label> </_object> <_content/> </org.openmdx.ui1.ElementDefinition></pre>
----------------	---

8.4.3 Fields / Attributes

With **order** (or more specifically **orderFieldGroup**) you can define the positioning of an object's attribute in the attribute pane. You must provide three values in the form of `<_item>xxx</item>` where the first value corresponds to the id of the Tab, the second value to the FieldGroup and the third value to the position within the respective FieldGroup.

The following example explains the placement of a contact's salutation:

The XML code defines the salutation attribute with the following structure:

```

<org.openmdx.ui.ElementDefinition name="org.opencrx.kernel.account1.Contact.salutation">
  <_object>
    <label>
      <_item>Salutation</item>
    </label>
    <order>
      <_item>0</item> <!-- Tab General -->
      <_item>0</item> <!-- FieldGroup 0 -->
      <_item>18</item> <!-- Position 18 -->
    </order>
  </_object>
</org.openmdx.ui.ElementDefinition>
    
```

The UI form displays the following attributes and their positions:

Attribute	Value	Attribute Position	FieldGroup
Salutation code:	Mr.	Attribute Position 10	FieldGroup 0
Salutation:		Attribute Position 18	FieldGroup 0
First Name:	Joe	Attribute Position 20	
Middle Name:	O	Attribute Position 30	
Last Name:	Guest	Attribute Position 40	
Suffix:		Attribute Position 50	
Nick Name:	jq	Attribute Position 60	
Alias name:	guest	Attribute Position 70	
Postal Address Home / Business			
Address line(s):	Mr. Joe Guest c/o CRIXP Corp.	Attribute Position 80	FieldGroup 10
Street / PO Box:	Technopark	Attribute Position 90	
State:		Attribute Position 100	
ZIP/Postal code:	8005	Attribute Position 100	
City:	Zürich	Attribute Position 120	
Country/Region:	Switzerland	Attribute Position 130	
Contact Information			
Preferred method:	E-Mail		FieldGroup 20
Spoken language:	German [de/ger/deu]	No postal mail:	<input checked="" type="checkbox"/>
Written language:	German [de/ger/deu]	No bulk postal mail:	<input checked="" type="checkbox"/>
No E-Mail:	<input type="checkbox"/>	Do not phone:	<input checked="" type="checkbox"/>
		Do not fax:	<input checked="" type="checkbox"/>
Job Information			
Job Title:	Director	Organization:	CRIXP Corp. FieldGroup 30
Job Role:	Chief Tester	Department:	Quality Assurance

Figure 11: Placement of attributes in the attribute pane (Tab, Group, Position)

With the tag `<columnBreak>>true</columnBreak>` you can place the current attribute in a new column within the given Field Group.

8.5 Grid Panes

Shared and composite associations as well as multi-valued references are mapped to grids. Such grids can be positioned by providing three values in the form of `<_item>xxx</item>` where the first value corresponds to the id of the Grid Pane and the third value to the position within the respective Pane (the second value is not used).

The following example explains the placement of an accounts composite association "member":

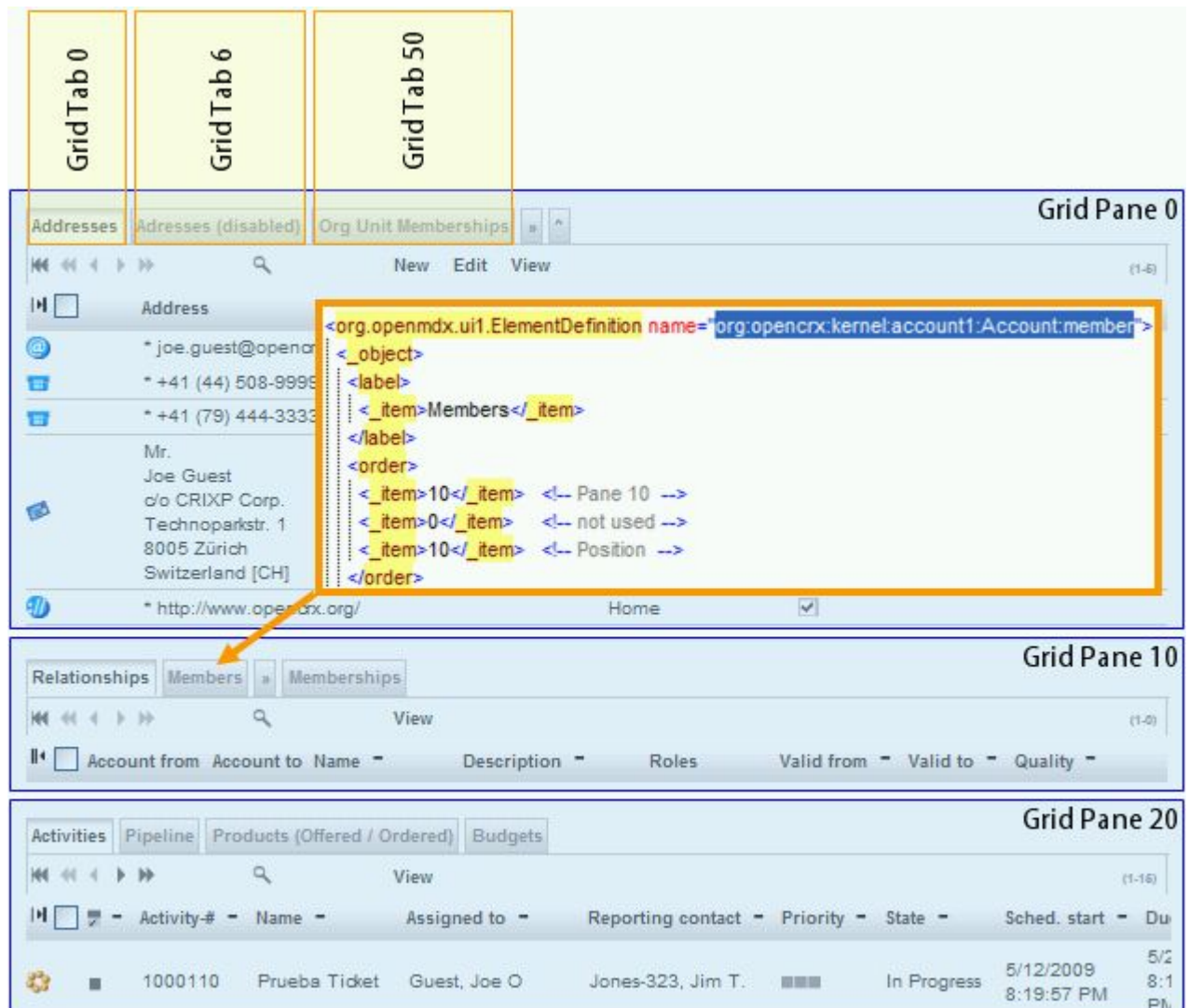


Figure 12: Placement of grid tabs (Pane, -, Position)

With **order** (or more specifically **orderObjectContainer**) you can define the positioning of an object's attribute within grids. You must provide three values in the form of `<_item>xxx</item>` where these values will be used to define an (alphabetical) order.

The following example explains the placement of a an address's usage:

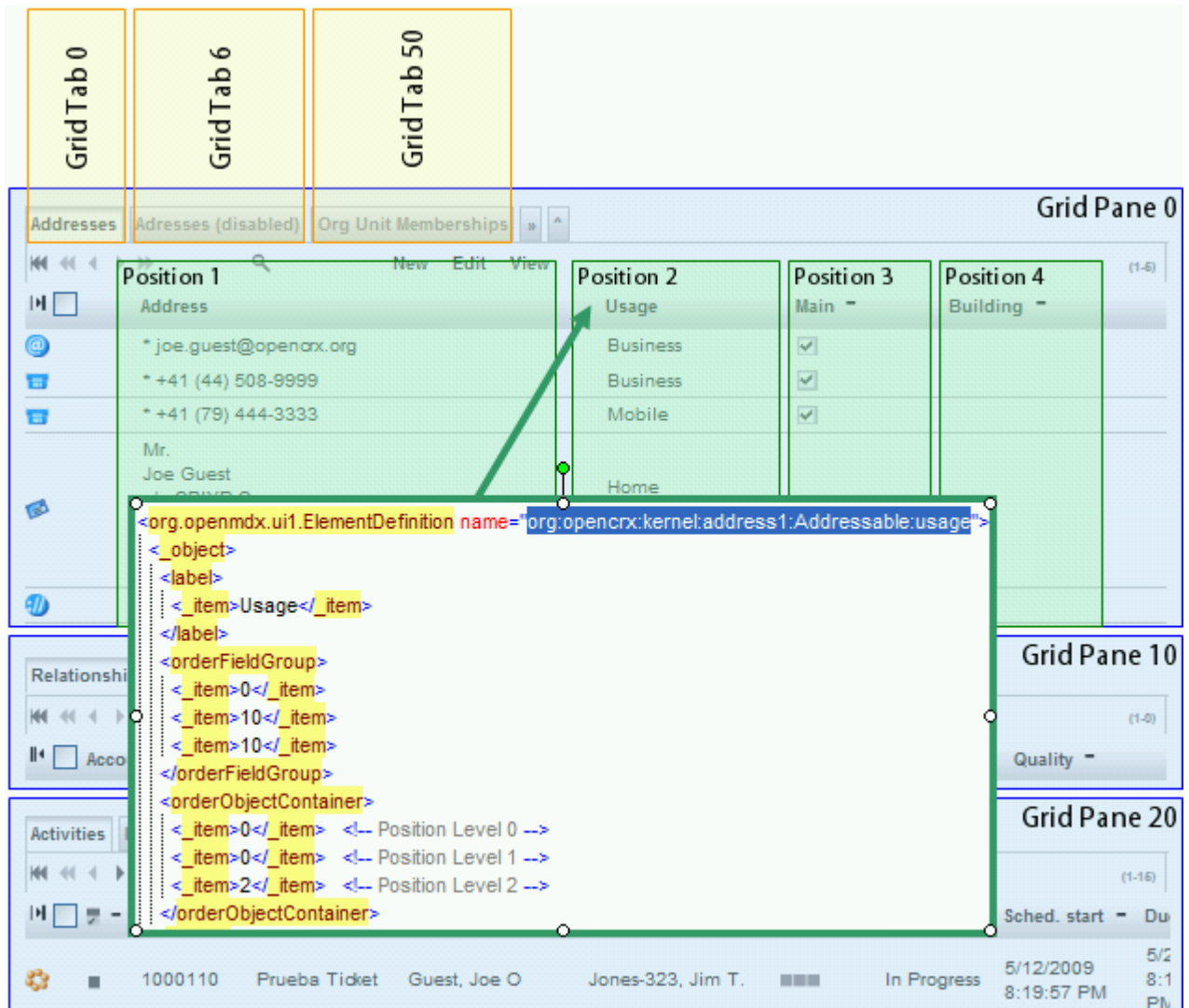


Figure 13: Placement of attributes in grids (Level 0, Level 1, Level 2)

8.5.1 Expanding/Collapsing Tabs

Grid Tabs are rendered collapsed if the label starts with a "»" character:

Example	<pre><org.openmdx.ui1.ElementDefinition name="org:opencrx:kernel:account1:Account:partner"> <_object> <active>true</active> <tooltip> <_item>Partners</_item> </tooltip> <label> <_item>>Partners</_item> </label> <order> <_item>0</_item> <_item>0</_item> <_item>97</_item> </order> </_object> </content/> </org.openmdx.ui1.ElementDefinition></pre>
---------	--

8.5.2 AdditionalElementDefinition

8.5.3 Base Filter for Grid Tabs

8.5.4 Selection of Filterable Attributes (maxMember)

8.5.5 Selection of Visible Attributes (showMaxMember)

8.5.6 Advanced Attribute Selection (showMemberRange)

8.6 Various XML Tags Explained

8.6.1 active

Purpose	Enable or disable a UI element. No default value.
Examples	<pre><active>true</active> <active>>false</active></pre>

8.6.2 backColor

Purpose	Set the background color of an attribute. No default value.
Examples	<pre><backColor>#FF0000</backColor> <backColor>inherit</backColor></pre>

8.6.3 changeable

Purpose	Enable updates or limit access to read-only. Default value is true.
Examples	<pre><changeable>true</changeable> <changeable>>false</changeable></pre>

8.6.4 color

Purpose	Set the (font) color of an attribute. No default value.
Examples	<pre><color>#FFFFFF</color></pre> <pre><color>inherit</color></pre>

8.6.5 columnBreak

Purpose	Add column break so that the current element is the first element of a new column within given field group. No default value.
Examples	<pre><columnBreak>true</columnBreak></pre> <pre><columnBreak>>false</columnBreak></pre>

8.6.6 defaultValue

Purpose	Default value of an attribute when a new object is created. No default value.
Examples	<pre><defaultValue>true</defaultValue></pre> <pre><defaultValue>7</defaultValue></pre>

8.6.7 eventHandler

8.6.8 filterable

Purpose	Enable or disable filtering for an attribute. While you can disable filtering on any attribute, enabling filtering makes only sense if the respective query is implemented. By default, filtering is not supported for references. Default value is true.
Examples	<pre><filterable>true</filterable></pre> <pre><filterable>>false</filterable></pre>

8.6.9 iconKey

Purpose	Allows you to set the name of the GIF file that gets displayed for a particular object type. Default value is class name.
Examples	<pre><iconKey>AccountAssignment</iconKey></pre> <pre><iconKey>Contact</iconKey></pre>

8.6.10 inPlace

Purpose	Enable or disable in place editing of a grid's objects. Default value is false.
Example	<pre><org.openmdx.ui1.ElementDefinition name="org:opencrx:kernel:product1:Product:relatedProduct"> <_object> <active>true</active> <inPlace>true</inPlace> <toolTip> <_item>Related Products</_item> </toolTip> <label> <_item>Related Products</_item> </label> <order> <_item>40</_item> <_item>0</_item> <_item>10</_item> </order> <maxMember>9</maxMember> <showMaxMember>4</showMaxMember> </_object> <_content/> </org.openmdx.ui1.ElementDefinition></pre>

8.6.11 label

Purpose	Allows you to set the label of an object. No default value.
Examples	<pre><label> <_item>Related Products</_item> </label></pre>

8.6.12 mandatory

Purpose	Make an attribute mandatory or optional. Default value is false for optional attributes (according to UML model) and true otherwise.
Examples	<pre><mandatory>true</mandatory></pre> <pre><mandatory>false</mandatory></pre>

8.6.13 min**Value**

Purpose	Define the minimum acceptable value of a numerical field. No default value.
Examples	<code><minValue>0</minValue></code> <code><minValue>-100</minValue></code>

8.6.14 max**Length**

Purpose	Limit the size of an input field (i.e. number of characters a user can enter). No default value.
Examples	<code><maxLength>5</maxLength></code> <code><maxLength>255</maxLength></code>

8.6.15 max**Member**

Purpose	Set the maximum number of attributes that are available in a grid (not necessarily visible, but at least filterable). The default value is 6.
Examples	<code><maxMember>0</maxMember></code> <code><maxMember>-100</maxMember></code>

8.6.16 max**Value**

Purpose	Define the maximum acceptable value of a numerical field. No default value.
Examples	<code><maxValue>0</maxValue></code> <code><maxValue>-100</maxValue></code>

8.6.17 order

8.6.18 orderFieldGroup

8.6.19 orderObjectContainer

8.6.20 showMaxMember

8.6.21 showMemberRange

8.6.22 skipRow

8.6.23 spanRow

8.6.24 sortable

8.6.25 toolTip

8.7 DataBindings

8.7.1 PropertyDataBinding

PropertyDataBindings provide you with access to a virtually unlimited pool of user-definable attributes to extend CrxObjects. From a performance point of view, PropertyDataBindings are not quite as efficient as the User-definable attributes of CrxObject because the latter are always stored in the same table as the extended object, whereas attributes based on PropertyDataBindings are stored in a separate table (leading to additional SELECT statements). Nevertheless, whenever you need additional attributes that cannot be mapped to user-definable attributes, you should still consider PropertyDataBindings before you jump at extending the openCRX core model.



Tip

If you need additional attributes, approach the issue as follows:

1. Map your requirements to User-definable attributes of CrxObject (see chapter 4.2.1).
2. If you run out of attributes, use PropertyDataBindings.
3. Only as a last resort should you consider extending the openCRX UML Model as such an extension requires expert knowledge and is not suitable for the faint-hearted.



Warning

Extending the openCRX UML Model is for experts only!

8.7.1.1 A comprehensive example with PropertyDataBindings

The following UI customizing file demonstrates how to extend the Product class with a set of attributes, one of each type. Deploy the file **zorder_product.xml** to the directory `{TOMCAT_INSTALL_DIR}/apps/openctx-core-CRX/openctx-core-CRX/WEB-INF/config/ui/Root/en_US` and then restart Tomcat.

Listing 7: UI Customizing File *zorder_product.xml*

```
<?xml version="1.0" encoding="UTF-8"?>
<org.openmdx.base.Authority xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="org.openmdx:u1"
xsi:noNamespaceSchemaLocation="xri:+resource/org/openmdx/u1/xmi/u1.xsd">
  <_object/>
  <_content>
    <provider>
      <org.openmdx.base.Provider qualifiedName="CRX" _operation="null">
        <_object/>
        <_content>
          <segment>
            <org.openmdx.u1.Segment qualifiedName="Root" _operation="null">
              <_object/>
              <_content>
                <featureDefinition>
```

```

    <org.openmdx.ui1.StructuralFeatureDefinition
qualifiedName="org:opencrx:kernel:product1:Product:Extension!myBoolean">
  <_object>
    <type>org:w3c:boolean</type>
    <multiplicity>0..1</multiplicity>
    <changeable>true</changeable>
  </_object>
  <_content/>
</org.openmdx.ui1.StructuralFeatureDefinition>

    <org.openmdx.ui1.StructuralFeatureDefinition
qualifiedName="org:opencrx:kernel:product1:Product:Extension!myInteger">
  <_object>
    <type>org:w3c:integer</type>
    <multiplicity>0..1</multiplicity>
    <changeable>true</changeable>
  </_object>
  <_content/>
</org.openmdx.ui1.StructuralFeatureDefinition>

    <org.openmdx.ui1.StructuralFeatureDefinition
qualifiedName="org:opencrx:kernel:product1:Product:Extension!myDecimal">
  <_object>
    <type>org:w3c:decimal</type>
    <multiplicity>0..1</multiplicity>
    <changeable>true</changeable>
  </_object>
  <_content/>
</org.openmdx.ui1.StructuralFeatureDefinition>

    <org.openmdx.ui1.StructuralFeatureDefinition
qualifiedName="org:opencrx:kernel:product1:Product:Extension!myString">
  <_object>
    <type>org:w3c:string</type>
    <multiplicity>0..1</multiplicity>
    <changeable>true</changeable>
  </_object>
  <_content/>
</org.openmdx.ui1.StructuralFeatureDefinition>

    <org.openmdx.ui1.StructuralFeatureDefinition
qualifiedName="org:opencrx:kernel:product1:Product:Extension!myDate">
  <_object>
    <type>org:w3c:date</type>
    <multiplicity>0..1</multiplicity>
    <changeable>true</changeable>
  </_object>
  <_content/>
</org.openmdx.ui1.StructuralFeatureDefinition>

    <org.openmdx.ui1.StructuralFeatureDefinition
qualifiedName="org:opencrx:kernel:product1:Product:Extension!myDateTime">
  <_object>
    <type>org:w3c:dateTime</type>
    <multiplicity>0..1</multiplicity>
    <changeable>true</changeable>
  </_object>
  <_content/>
</org.openmdx.ui1.StructuralFeatureDefinition>

    <org.openmdx.ui1.StructuralFeatureDefinition
qualifiedName="org:opencrx:kernel:product1:Product:Extension!myReference">
  <_object>
    <type>org:opencrx:kernel:account1:Account</type>
    <multiplicity>0..1</multiplicity>
    <changeable>true</changeable>
  </_object>
  <_content/>
</org.openmdx.ui1.StructuralFeatureDefinition>

  </featureDefinition>

  <elementDefinition>

    <org.openmdx.ui1.ElementDefinition name="org:opencrx:kernel:product1:Product:Pane:Attr:Tab:0:Group:5">
  <_object>
    <active>true</active>
    <toolTip>
      <_item/>
    </toolTip>
    <label>
      <_item>Extended Attributes (PropertyDataBinding)</_item>
    </label>
  </_object>
</org.openmdx.ui1.ElementDefinition>

```

```

</_object>
<_content/>
</org.openmdx.ui1.ElementDefinition>

<org.openmdx.ui1.ElementDefinition name="org:opencrx:kernel:product1:Product:Extension!myBoolean">
  <_object>
    <dataBindingName>org.opencrx.kernel.portal.BooleanPropertyDataBinding</dataBindingName>
    <active>true</active>
    <tooltip>
      <_item>My boolean</_item>
    </tooltip>
    <label>
      <_item>My boolean</_item>
    </label>
    <order>
      <_item>0</_item>
      <_item>5</_item>
      <_item>10</_item>
    </order>
  </_object>
<_content/>
</org.openmdx.ui1.ElementDefinition>

<org.openmdx.ui1.ElementDefinition name="org:opencrx:kernel:product1:Product:Extension!myInteger">
  <_object>
    <dataBindingName>org.opencrx.kernel.portal.IntegerPropertyDataBinding</dataBindingName>
    <active>true</active>
    <tooltip>
      <_item>My integer</_item>
    </tooltip>
    <label>
      <_item>My integer</_item>
    </label>
    <order>
      <_item>0</_item>
      <_item>5</_item>
      <_item>20</_item>
    </order>
  </_object>
<_content/>
</org.openmdx.ui1.ElementDefinition>

<org.openmdx.ui1.ElementDefinition name="org:opencrx:kernel:product1:Product:Extension!myDecimal">
  <_object>
    <dataBindingName>org.opencrx.kernel.portal.DecimalPropertyDataBinding</dataBindingName>
    <active>true</active>
    <decimalPlaces>4</decimalPlaces>
    <hasThousandsSeparator>true</hasThousandsSeparator>
    <tooltip>
      <_item>My decimal</_item>
    </tooltip>
    <label>
      <_item>My decimal</_item>
    </label>
    <order>
      <_item>0</_item>
      <_item>5</_item>
      <_item>30</_item>
    </order>
  </_object>
<_content/>
</org.openmdx.ui1.ElementDefinition>

<org.openmdx.ui1.ElementDefinition name="org:opencrx:kernel:product1:Product:Extension!myString">
  <_object>
    <dataBindingName>org.opencrx.kernel.portal.StringPropertyDataBinding</dataBindingName>
    <active>true</active>
    <tooltip>
      <_item>My string</_item>
    </tooltip>
    <label>
      <_item>My string</_item>
    </label>
    <order>
      <_item>0</_item>
      <_item>5</_item>
      <_item>40</_item>
    </order>
  </_object>
<_content/>
</org.openmdx.ui1.ElementDefinition>

```

```

<org.openmdx.ui1.ElementDefinition name="org:opencrx:kernel:product1:Product:Extension!myDate">
  <_object>
    <dataBindingName>org.opencrx.kernel.portal.DatePropertyDataBinding</dataBindingName>
    <active>true</active>
    <toolTip>
      <_item>My date</_item>
    </toolTip>
    <label>
      <_item>My date</_item>
    </label>
    <order>
      <_item>0</_item>
      <_item>5</_item>
      <_item>50</_item>
    </order>
  </_object>
</_content/>
</org.openmdx.ui1.ElementDefinition>

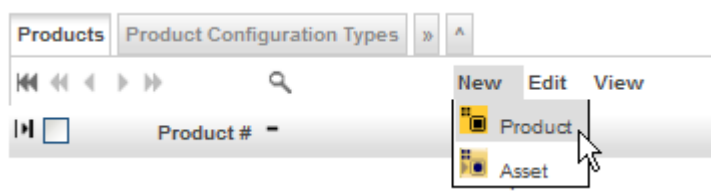
<org.openmdx.ui1.ElementDefinition name="org:opencrx:kernel:product1:Product:Extension!myDateTime">
  <_object>
    <dataBindingName>org.opencrx.kernel.portal.DateTimePropertyDataBinding</dataBindingName>
    <active>true</active>
    <toolTip>
      <_item>My dateTime</_item>
    </toolTip>
    <label>
      <_item>My dateTime</_item>
    </label>
    <order>
      <_item>0</_item>
      <_item>5</_item>
      <_item>60</_item>
    </order>
  </_object>
</_content/>
</org.openmdx.ui1.ElementDefinition>

<org.openmdx.ui1.ElementDefinition name="org:opencrx:kernel:product1:Product:Extension!myReference">
  <_object>
    <dataBindingName>org.opencrx.kernel.portal.ReferencePropertyDataBinding</dataBindingName>
    <active>true</active>
    <toolTip>
      <_item>My reference (to Account)</_item>
    </toolTip>
    <label>
      <_item>My reference (to Account)</_item>
    </label>
    <order>
      <_item>0</_item>
      <_item>5</_item>
      <_item>70</_item>
    </order>
  </_object>
</_content/>
</org.openmdx.ui1.ElementDefinition>
</elementDefinition>

</_content>
</org.openmdx.ui1.Segment>
</segment>
</_content>
</org.openmdx.base.Provider>
</provider>
</_content>
</org.openmdx.base.Authority>

```

Once Tomcat is up and running, navigate to the tab **Products** and create a new product as shown below:



The screen for entering new products should now include a new field group labeled **Extended Attributes (PropertyDateBinding)** containing additional fields as shown below:

The screenshot shows a web form titled "Untitled - Product" with tabs for "General", "Details", and "System". The "General" tab is active. The form contains several input fields:

- Name: [text input]
- Description: [text input]
- Product status: [dropdown menu, value: N/A]
- Product number: [text input]
- Active/Valid from: [text input]
- Expires on: [text input]
- Min positions: [text input]
- Max positions: [text input]
- Default positions: [text input]
- Configuration type: [text input]
- Min quantity: [text input]
- Max quantity: [text input]
- Offset quantity: [text input]
- Default quantity: [text input]
- [min..max] handling: [dropdown menu, value: [0] N/A]

A red box highlights the "Extended Attributes (PropertyDateBinding)" section, which includes:

- My boolean: [checkbox]
- My integer: [text input]
- My decimal: [text input]
- My string: [text input]
- My date: [text input]
- My dateTime: [text input]
- My reference (to Account): [text input with search icon]

At the bottom, there is a "Detailed description" field and "Save" and "Cancel" buttons.

Figure 14: Sample Extension of class Product with PropertyDataBindings

Enter some data as follows:

- Name: **Sample Product**
- Product number: **P-1234**
- My boolean: **check it**
- My integer: **123**
- My decimal: **987654321.0123**
- My string: **sample string**
- My date: **1/23/2009**
- My dateTime: **1/23/2009 12:34:56 AM**
- My reference (to Account): **point to guest**

Save to new product and then navigate to it.

Expand the grid tabs in the first grid pane by clicking on the tab [**>>**]. You will see additional tabs like [**Notes**], [**Folders**], etc. Click on the tab [**Property Sets**] to see the entry Extension as shown below:

The screenshot shows the 'Sample Product / P-1234 - Product' form. The 'Property Sets' tab is selected, displaying a table of extended attributes. The table includes fields like 'My boolean' (checked), 'My integer' (123), 'My decimal' (987,654,321.0123), 'My string' (sample string), 'My date' (1/23/2009), 'My dateTime' (1/23/2009 12:34:56 AM), and 'My reference (to Account)' (Guest). Below the table is a 'Detailed description' field. The bottom of the screen shows a grid pane with a tab labeled 'Extension'.

Name:	Sample Product	Product number:	P-1234
Description:		Active/Valid from:	
Product status:	N/A	Expires on:	
Min positions:		Default positions:	
Max positions:			
Configuration type:			
Min quantity:		Default quantity:	
Max quantity:		[min..max] handling:	[0] N/A
Offset quantity:			
Extended Attributes (Property/DataBinding)			
My boolean:	<input checked="" type="checkbox"/>		
My integer:	123		
My decimal:	987,654,321.0123		
My string:	sample string		
My date:	1/23/2009		
My dateTime:	1/23/2009 12:34:56 AM		
My reference (to Account):	Guest,		
Detailed description:			

Figure 15: Sample Extension of class Product with PropertyDataBindings

The PropertySet Extension was created automatically when you saved the new product. It is called **Extension** because that is what was customized in the file **zorder_product.xml**. The PropertySet Extension contains all the attributes defined with PropertyDataBindings, i.e. *myBoolean*, *myInteger*, *myDecimal*, *myString*, *myDate*, *myDateTime*, and *myReference*. Let's verify this by clicking on the icon of the respective Property Set.

You should see the following:

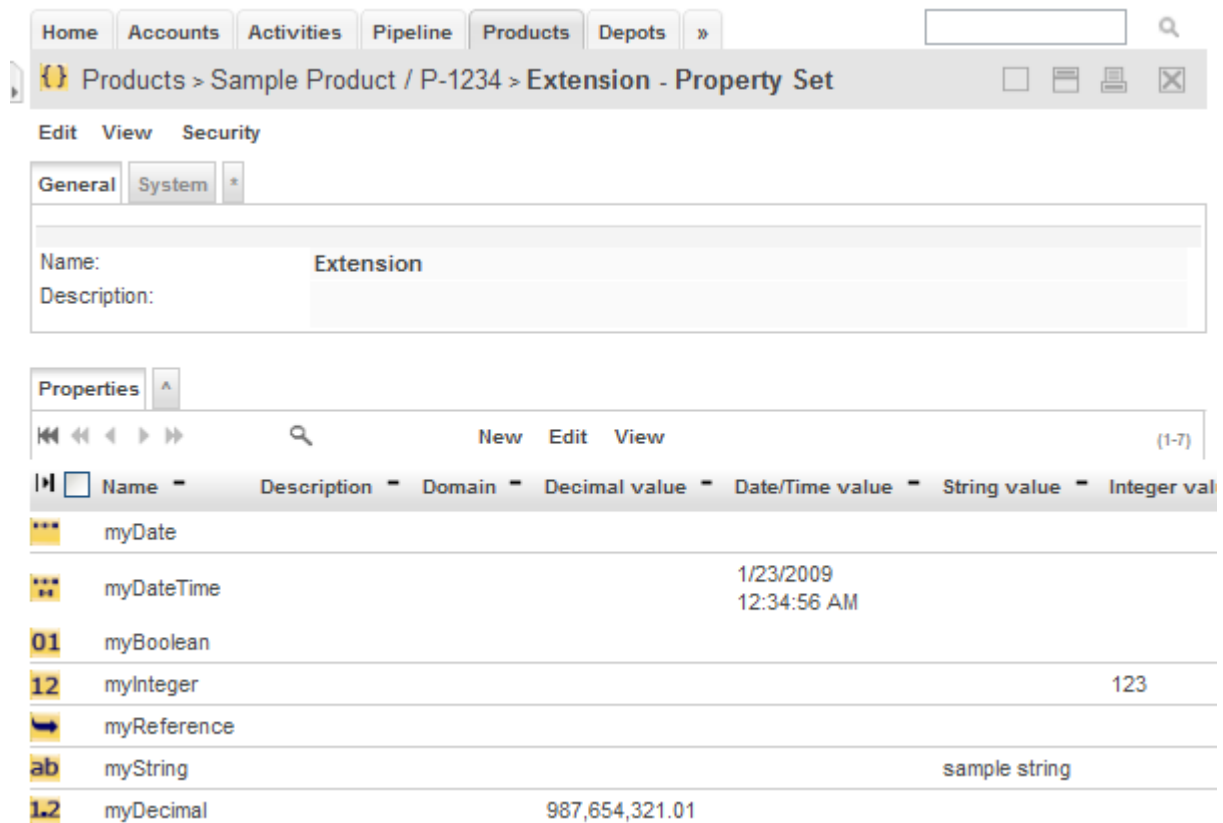


Figure 16: Property Set Extension

While all of these properties could be edited right within the Property Set, it is definitely more comfortable (and natural) to edit them within the attribute pane of products.

Note that this extension of the class Product did not require any modifications of the openCRX UML Model. We did not even have to modify the database schema. The whole extension is done with customizing features only!

8.7.1.2 BooleanPropertyDataBinding**8.7.1.3 IntegerPropertyDataBinding****8.7.1.4 DecimalPropertyDataBinding****8.7.1.5 StringPropertyDataBinding****8.7.1.6 DatePropertyDataBinding****8.7.1.7 DateTimePropertyDataBinding****8.7.1.8 ReferencePropertyDataBinding**

8.7.2 DataBinding ProductConfigurationSet

8.7.3 DataBinding ProductConfigurationTypeSet

8.7.4 DataBindings with Referenced Objects

8.7.5 DataBindings with Composite Objects

8.7.6 DataBindings for Addresses

8.7.6.1 EmailAddressDataBinding

8.7.6.2 PhoneNumberDataBinding

8.7.6.3 WebAddressDataBinding

8.7.7 DataBindings Example Grid:

```

<featureDefinition>
  <org.openmdx.ui1.StructuralFeatureDefinition
    qualifiedName="org:opencrx:kernel:home1:UserHome:tracker!${USER}~Private!filteredActivity">
    <type>org:opencrx:kernel:activity1:Activity</type>
    <multiplicity>0..n</multiplicity>
    <changeable>>false</changeable>
    <isReference>true</isReference>
  </org.openmdx.ui1.StructuralFeatureDefinition>
</featureDefinition>

<org.openmdx.ui1.ElementDefinition name="org:opencrx:kernel:home1:UserHome:tracker!${USER}~Private!
filteredActivity">
  <_object>
    <active>true</active>
    <changeable>>false</changeable>
    <toolTip>
      <_item>Private Activities</_item>
    </toolTip>
    <label>
      <_item>Private Activities</_item>
    </label>
    <order>
      <_item>0</_item>
      <_item>0</_item>
      <_item>3</_item>
    </order>
    <maxMember>5</maxMember>
    <dataBindingName>org.opencrx.kernel.portal.FilteredActivitiesDataBinding</dataBindingName>
  </_object>
</_content/>
</org.openmdx.ui1.ElementDefinition>

```

8.7.8 DataBinding AssignedActivityGroupsDataBinding

Listing 8: Example AssignedActivityGroupsDataBinding

```
<featureDefinition>
  <org.openmdx.ui1.StructuralFeatureDefinition
qualifiedName="org:opencrx:kernel:activity1:Activity:assignedGroupName">
  <_object>
    <type>org:opencrx:kernel:activity1:ActivityGroup</type>
    <multiplicity>list</multiplicity>
    <changeable>false</changeable>
  </_object>
  <_content/>
</org.openmdx.ui1.StructuralFeatureDefinition>
</featureDefinition>

<org.openmdx.ui1.ElementDefinition name="org:opencrx:kernel:activity1:Activity:assignedGroupName">
  <_object>
    <dataBindingName>org.opencrx.kernel.portal.AssignedActivityGroupsDataBinding?type=tracker</dataBindingName>
    <active>true</active>
    <spanRow>6</spanRow>
    <toolTip>
      <_item>Assigned Trackers</_item>
    </toolTip>
    <label>
      <_item>Assigned Trackers</_item>
    </label>
    <shortLabel>
      <_item>Groups</_item>
    </shortLabel>
    <orderFieldGroup>
      <_item>10</_item>
      <_item>100</_item>
      <_item>10</_item>
    </orderFieldGroup>
    <orderObjectContainer>
      <_item>0</_item>
      <_item>0</_item>
      <_item>9</_item>
    </orderObjectContainer>
  </_object>
  <_content/>
</org.openmdx.ui1.ElementDefinition>
```

Optional Parameter type:

Syntax: ?type={ [tracker|milestone|category] ", " }

If you omit the optional parameter type all assigned activity groups are shown. The above example selects ActivityTrackers only. You can also provide a comma-separated list of types, e.g. ?type=tracker,milestone.

8.7.9 DataBinding DocumentFolderAssignmentsDataBinding

Optional Parameter name:

Syntax: ?name={ name prefix ", " }

If you omit the optional parameter type all document folder assignments are shown.

8.7.10 DataBinding FilteredActivitiesDataBinding

8.7.11 DataBinding FormattedNoteDataBinding

9 Code Table XML Files

9.1 Overview

9.2 Code Table Overloading

9.2.1 Adding Codes to Existing Code Tables

9.2.2 Disabling Existing Codes

9.2.3 Replacing Existing Code Tables

9.3 Segment-Specific Code Tables

10 Groovy Controls

10.1 MenuOps – openCRX Operations Menu

10.2 Navigation – openCRX Breadcrum

10.3 North – openCRX Header

10.4 RootMenu – openCRX Top Level Tabbed Menu

10.5 RootPanel – openCRX Top Level PopUp Menu

10.6 Search – openCRX Index-based Search

11 JSP Wizards

See the various examples in the directory `...\opencrx-core-CRX\wizards\en_US`

12 Forms

Forms were introduced with openCRX v2.4. Have a look at the following examples included in the distribution:

- Form definitions are located in the directory `...\opencrx-core-CRX\WEB-INF\config\ui\Root\en_US`
- Form wizards are located in the directory `...\opencrx-core-CRX\wizards\en_US`

Form	Wizards making use of the Form
CreateActivityForm.xml	CreateActivityWizard.jsp
CreateActivityTrackerForm.xml	CreateAgendaWizard.jsp CreateBugTrackerWizard.jsp CreateCampaignWizard.jsp CreateProjectWizard.jsp
CreateContactForm.xml	CreateContactWizard.jsp
CreateContractForm.xml	CreateContractWizard.jsp
CreateLeadForm.xml	CreateLeadWizard.jsp
CreateProductForm.xml	CreateProductWizard.jsp
ScheduleEventForm.xml	ScheduleEventWizards.jsp

13 Layout JSPs

openCRX is distributed with 2 default layout JSPs located in the directory `...\opencrx-core-CRX\WEB-INF\config\layout\en_US`.

13.1 show-Default.jsp

This layout JSP renders all pages that show information (typically an Inspector containing information about the current object and all the grids containing associated information). This layout JSP is generic and can handle any object. It is provided by openMDX/portal.

13.2 edit-Default.jsp

Similarly, this layout JSP renders all pages that are used to edit objects and create new objects. This layout JSP is generic and can handle any object. It is provided by openMDX/portal.

13.3 Custom Layout JSPs

If you have a need for specialized screens for a particular object in edit and/or show mode, you can write your own layout JSP and deploy it to the above-mentioned directory. The file name of your custom layout JSP determines which objects (or rather: objects of which class) will be handled by your custom layout JSP.

Example:

Let's assume you want to replace the default edit screen for openCRX Contacts (i.e. class **org.opencrx.kernel.account1.Contact**) with a custom layout JSP. Name your file

edit-org.opencrx.kernel.account1.Contact.jsp

and deploy it to the directory `...\WEB-INF\config\layout\en_US`. After restarting Tomcat or your application server your new layout JSP will be active.



Tip

If you develop localized JSPs you can create new directories for the respective locales and then deploy your localized JSPs there. The fallback algorithms are comparable to those in ui customization.



Important

You must provide a layout JSP for the base locale, by default the locale en_US.

14 Advanced Customizing Options

14.1 Workflows

14.2 Java Controls

14.3 Portal Extensions

14.4 UML Model Extensions

14.5 Application Logic Extensions

15 Other Customizing Options

A collection of useful configuration and customizing options.

15.1 web.xml

All of the following settings can be made in the file

`...\opencrx-core-CRX\WEB-INF\web.xml`

15.1.1 uiRefreshRate

By default, the UI configuration is loaded at startup only. If you are changing customizing files during a customization session you might want to reload the UI periodically to verify your changes without restarting the servlet container. You can do so by setting the respective value to something else than 0. The refresh period is measured in milliseconds, i.e. the following example would cause the UI configuration to reload once per minute:

Listing 9: uiRefreshRate in web.xml

```
...
<!-- ui refresh rate -->
<init-param>
  <param-name>uiRefreshRate</param-name>
  <param-value>60000</param-value>
</init-param>
...
```

15.1.2 load-on-startup

By default, the ObjectInspectorServlet is initialized with the first login (and hence the first login can take some time). If you prefer to initialize the ObjectInspectorServlet right after the deployment of the respective EAR, uncomment the following section in web.xml:

Listing 10: load-on-startup in web.xml

```
...
<!-- activate if ObjectInspectorServlet should be initialized at startup -->
<load-on-startup>1</load-on-startup>
...
```

15.1.3 session-timeout

By default, the session timeout is set to 30 minutes. If you want to shorten or lengthen the session timeout you can do so by changing the value of session-timeout (measured in minutes) in web.xml. The following example shows how to change the timeout to 2 hours:

Listing 11: session-timeout in web.xml

```
...
<session-config>
  <session-timeout>120</session-timeout>
</session-config>
...
```

15.1.4 transport-guarantee

If you want to force your users to connect with SSL you can require a transport guarantee and replace the default value NONE with CONFIDENTIAL. The following example shows how to require the application admin (admin-Root) to connect with SSL:

Listing 12: transport-guarantee in web.xml

```
...
<security-constraint id="security_constraint-Root">
  <web-resource-collection id="c-Root">
    <web-resource-name>c-Root</web-resource-name>
    <url-pattern>/LogConsoleServlet/*</url-pattern>
    <url-pattern>/WorkflowController/*</url-pattern>
    <http-method>GET</http-method>
    <http-method>POST</http-method>
  </web-resource-collection>
  <auth-constraint id="auth_constraint-Root">
    <role-name>OpenCrxRoot</role-name>
  </auth-constraint>
  <user-data-constraint id="user_data_constraint-Root">
    <transport-guarantee>CONFIDENTIAL</transport-guarantee>
  </user-data-constraint>
</security-constraint>
...
```

16 Next Steps

You might want to have a look at some of the additional documentation published at <http://www.opencrx.org/documents.htm>. The openCRX Admin Guide might be of particular interest to you.